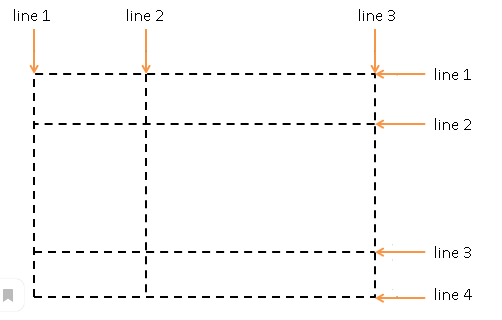
**Практическая работа № 5 Тема «Позиционирование элементов в Grid»**

**Цель: научиться позиционировать элементы в Grid Css**

**Теория**

Грид представляет собой набор ячеек, которые образуются на пересечении столбцов и строк. Но сами строки и столбцы образуются с помощью grid-линий, которые рассекают грид по вертикали и горизонтали:



И по умолчанию каждый элемент в гриде позиционируется в одну ячейку по порядку. Но мы можем более точно настроить расположение элемента в гриде с помощью ряда свойств:

* grid-row-start: задает начальную горизонтальную grid-линию, с которой начинается элемент
* grid-row-end: указывает, до какой горизонтальной grid-линии надо растягивать элемент
* grid-column-start: задает начальную вертикальную grid-линию, от которой начинается элемент
* grid-column-end: указывает, до какой вертикальной grid-линии нужно растягивать элемент

Например, растянем элемент на несколько столбцов:

html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width" />

<title>Grid Layout в CSS3</title>

<style>

.grid-container {

border: solid 2px #000;

display: grid;

grid-template-columns: repeat(4, 1fr);

grid-template-rows: repeat(3, 5em);

}

.special-item {

grid-column-start: 2;

grid-column-end: 5;

}

.grid-item {

text-align: center;

font-size: 1.1em;

padding: 1.5em;

color: white;

}

.color1 {

background-color: #675ba7;

}

.color2 {

background-color: #9bc850;

}

.color3 {

background-color: #a62e5c;

}

.color4 {

background-color: #2a9fbc;

}

.color5 {

background-color: #4e342e;

}

</style>

</head>

<body>

<div class="grid-container">

<div class="grid-item color1">Grid Item 1</div>

<div class="grid-item color2 special-item">Grid Item 2</div>

<div class="grid-item color3">Grid Item 3</div>

<div class="grid-item color4">Grid Item 4</div>

<div class="grid-item color5">Grid Item 5</div>

<div class="grid-item color1">Grid Item 6</div>

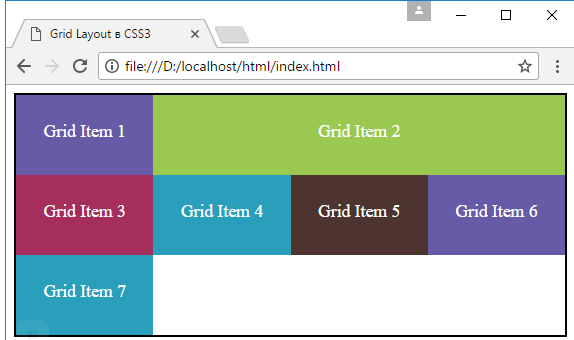
<div class="grid-item color4">Grid Item 7</div>

</div>

</body>

</html>

**1 Пример Грид на 7 эл. Различн размеры блоков**



Здесь определено четыре столбца, причем второму элементу присвоен особый класс special-item, который располагается, начиная со 2 grid-линии или 2-го столбца (grid-column-start: 2) до 5-й вертикальной grid-линии (grid-column-end: 5).

Второй элемент необязательно должен начинаться со второго столбца, это может быть любой другой: и 1-й, и 3-й и т.д. Например, если мы поместим второй элемент, начиная с 3-го столбца, то на месте второго столбца образуется пустота:

.special-item {

grid-column-start: 3;

grid-column-end: 5;

}



Вместо использования двух выше рассмотренных свойств мы можем использовать одно свойство - grid-column, которое принимает значения grid-column-start и grid-column-end через слеш:

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

Например, мы можем сократить стиль класса special-item следующим образом:

.special-item {

grid-column: 3 / 5;

}

Аналогично с помощью свойств grid-row-start и grid-row-end можно задать позиционирование элемента на несколько строк. Так, изменим класс special-itemследующим образом:

.special-item {

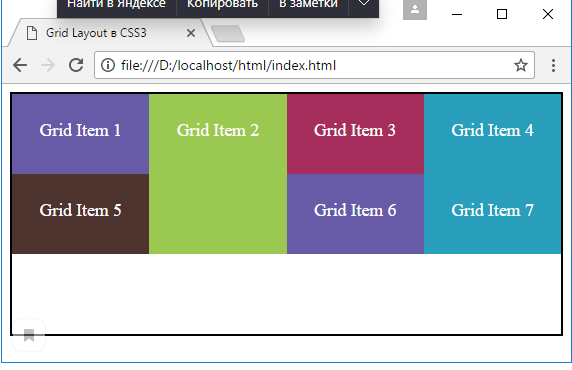
grid-column-start: 2;

grid-row-start: 1;

grid-row-end: 3;

}

В данном случае второй элемент позиционируется во втором столбце первой строки и растягивается до 3-й строки.



Вместо использования пары свойств grid-row-start и grid-row-end можно использовать одно общее свойство grid-row:

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

Так, мы могли бы изменить стиль special-item следующим образом:

.special-item {

grid-column-start: 2;

grid-row: 1 / 3;

}

span[¶](https://xsltdev.ru/css/grid-guide/grid-6/" \l "span" \o "Permanent link)

С помощью специального слова span можно задать растяжение элемента на несколько ячеек. После слова span указывается, на какое количество ячеек надо растянуть элемент:

.special-item {

grid-row: 1 / span 2;

grid-column: 2 / span 2;

}

Элемент помещается в ячейку, которая находится на пересечении первой строки и второго столбца, и растягивается на две строки вниз и на два столбца вправо.

grid-area[¶](https://xsltdev.ru/css/grid-guide/grid-6/" \l "grid-area" \o "Permanent link)

Свойство grid-area объединяет свойства grid-column и grid-row, позволяя сократить их запись:

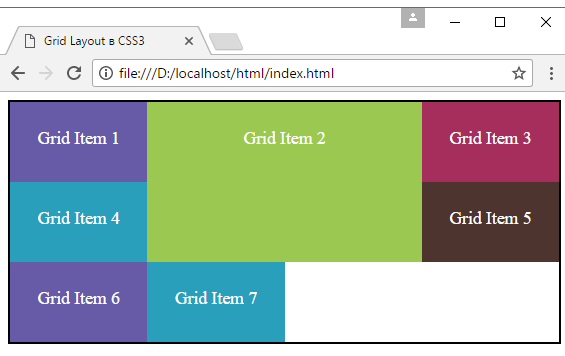
grid-area: row-start / column-start / row-end / column-end;

Например, изменим стили класса special-item:

.special-item {

grid-area: 1 / 2 / 3 / 4;

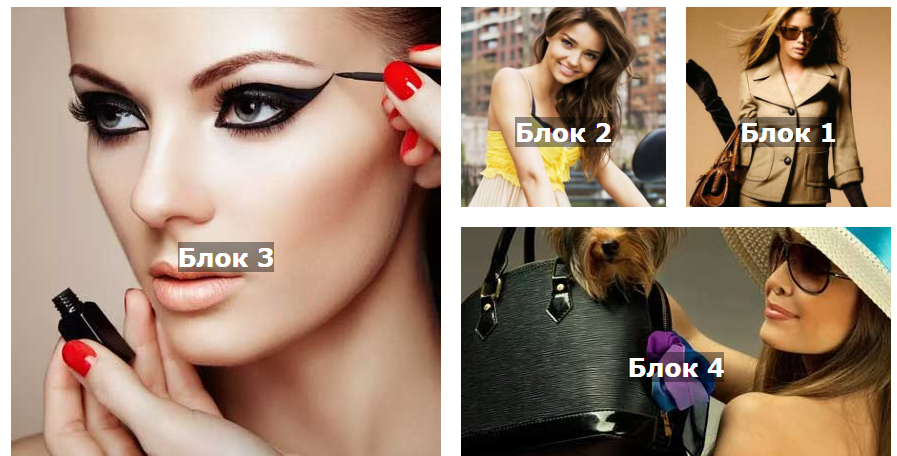
}



**Пример 2**

**Делаем верстку на модуле CSS Grid**

Макет



Прописываем блок с классом .grid-container, в него вложим блок с классом .imageBox, в котором будет находиться картинка и изображение. Таких блоков делаем четыре штуки и разворачиваем структуру.

1. .grid-container>(.imageBox>img[src="img/$.jpg"]+h3>{Блок $})\*4

Прописываем изображение и заголовки.

1. <div class="grid-container">
2. <div class="imageBox"><img src="img/1.jpg" alt="">
3. <h3>Блок 1</h3>
4. </div>
5. <div class="imageBox"><img src="img/2.jpg" alt="">
6. <h3>Блок 2</h3>
7. </div>
8. <div class="imageBox"><img src="img/3.jpg" alt="">
9. <h3>Блок 3</h3>
10. </div>
11. <div class="imageBox"><img src="img/4.jpg" alt="">
12. <h3>Блок 4</h3>
13. </div>
14. </div>

Это и есть вся структура, для реализации сетки на **модуле Grid**. Она логична, понятна, и в ней присутствует основной контейнер, в котором вложены его элементы.

Далее создадим файл стилей, сброшу все отступы и делаю сверху отступ в 30 пик.

1. body {
2. margin: 0;
3. padding: 0;
4. font-family: Verdana, Geneva, sans-serif;
5. }
6. .dws-wrapper {
7. margin-top: 30px;
8. }

И для того что бы начать пользоваться мощным инструментом CSS Grid, нужно назначить его основной контейнер. Напишем блоку с классом grid\_container display: grid.

1. .grid-container {
2. display: grid;
3. }

Вторым этапом определим шаблон сетки, которые задаются при помощи свойства grid-area, но что бы его задать, нужно для элементов Grid прописать индивидуальные имена, при помощи классов или как сделаем мы, воспользуемся псевдоклассом nth-child где в скобках укажем порядковый номер блока и при помощи grid-area зададим для каждой ячейки индивидуальное имя.

1. .grid-container div:nth-child(1) {
2. grid-area: box-1;
3. }
4. .grid-container div:nth-child(2) {
5. grid-area: box-2;
6. }
7. .grid-container div:nth-child(3) {
8. grid-area: box-3;
9. }
10. .grid-container div:nth-child(4) {
11. grid-area: box-4;
12. }

Затем зададим для блока **Grid**максимальную ширину в 900 пик. Почему в 900! Так как картинки по ширине занимают 430 пик, в сумме они будут 860 пик, и по 10 пик зададим у них отступы с каждой из сторон, выходит 40 пик., а полностью в сумме выйдет 900 пик.

Выравниваем при помощи margin: auto, далее описываем сетку.

1. .grid-container {
2. …
3. max-width: 900px;
4. margin: 0 auto;
5. }

Временно пока скрою текстовый блок, для этого отбираю его и задаю свойство display: none.

1. .imageBox h3 {
2. display: none;
3. }

Далее формируем свою сетку, прописываем grid-template-areas, и далее, используя имена блоков, которые задавали ранее, указываю в каком порядке, сколько ячеек будут они занимать.

1. .grid-container {
2. ….
3. grid-template-areas:
4. "box-1 box-2 box-3"
5. "box-1 box-4 box-4"
6. }

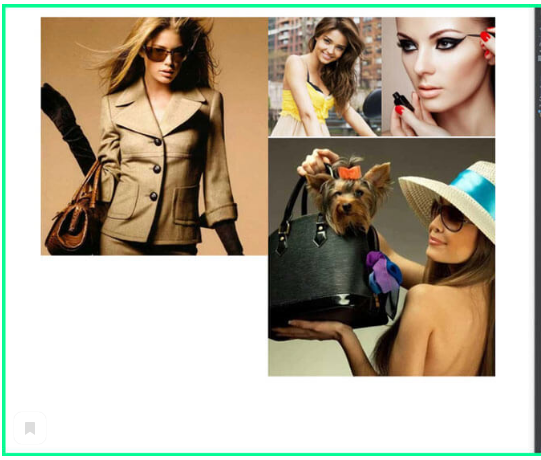
Элементы **Grid контейнера** выстроились, но они еще не знаю, сколько отведено для них пространство. Точнее они занимают определенное пространство ширины и высоты картинки, но нас это не устраивает. У блоков размеры картинок разные, и что бы нам индивидуально под каждый блок не нарезать картинку, сделаем это все при помощи CSS.

И давайте зададим 100% ширину картинки, что бы она занимала все пространство блока.

1. .imageBox img {
2. width: 100%;
3. }

Затем для каждой ячейки зададим свой размер ширины, для этого воспользуемся свойством grid-template-columns и каждой колонке задаем определенный размер. Первой сделаем 2 фракции, и двум остальным по одной фракции.

1. .grid-container {
2. …
3. grid-template-columns: 2fr 1fr 1fr;
4. }



Как видим, четвертый блок вылез за пределы основного контейнера. И что бы скрыть его часть, нужно для контейнера imagBox прописать overflow: hidden.

1. .imageBox {
2. overflow: hidden;
3. }

И данный стиль бы сработал, но из-за того что нет высоты контейнера, он не знает его границы. А какая должна быть высота? Обратимся к картинке, размер ее по высоте 450 пик., делим на пополам получается 225 пик.

При помощи grid-template-rows прописываем каждой строке высоту по 225 пик.

1. .grid-container {
2. …
3. grid-template-rows: 225px 225px;
4. }

Ну и для отступов можно воспользуемся свойством grid-gap, где каждой из сторон зададим по 10 пик, но я использую margin в 10 пик, это задаст сразу отступы в дальнейшем для мобильных устройств.

1. .imageBox {
2. …
3. margin: 10px;
4. }
6. .grid-container {
7. display: grid;
8. max-width: 900px;
9. margin: 0 auto;
10. grid-template-columns: 2fr 1fr 1fr;
11. grid-template-rows: 225px 225px;
12. grid-template-areas:
13. "box-1 box-2 box-3"
14. "box-1 box-4 box-4"
15. }

В принципе с контейнером закончили работать, и переходи к его элементам. Блоку с классом .imageBox прописываем position: relative, делаем его flex контейнером.

Для заголовка пропишем позиционирование абсолютное блоку, увеличим до 25 пик, сделаем прозрачный фон, и назначим белый цвет.

1. .imageBox h3{
2. position: absolute;
3. font-size: 25px;
4. background: rgba(0,0,0, 0.4);
5. color: #ffffff;
6. }

И для его выравнивания по горизонтали, зададим .imegeBox justify-content: center, а для вертикали align-items: center.

Элементы выровняли и далее можно переходить к адаптивности сетки.

**Адаптируем сетку на модуле CSS Grid**

Создаем дополнительный файл, он у нас подключается в head при разрешении менее 920 пик. Делаем проверку, убедились что он подключается и поехали дальше.

1. body {
2. background: red;
3. }

При таком разрешении делаем двух колоночную сетку, первым делом зададим допустимую максимальную ширину в 730 пик. Зададим ширину колонок по 1 фракции, а ширину ряда по 370 пик.

1. .grid-container {
2. max-width: 730px;
3. grid-template-columns: 1fr 1fr;
4. grid-template-rows: 370px;
5. grid-template-areas:
6. "box-1 box-2"
7. "box-3 box-4"
8. }

На разрешении менее 768 пик, делаем максимальную допустимую ширину 450 пик, для высоты и ширены ячейки задаем 1 фракцию, и делаем размету в одну ячейку.

1. @media only screen and (max-width: 768px) {
2. .grid-container {
3. max-width: 450px;
4. grid: 1fr / 1fr;
5. grid-template-areas:
6. "box-1"
7. "box-2"
8. "box-3"
9. "box-4"
10. }
11. }

И последний штрих, что бы блоки не сжимались мене 150 пик, так как это уже не красиво смотрится. Создаем медиа запрос от 280 пик, и для блоков пропишем минимальную допустимую ширину в 150 пик.

1. @media only screen and (max-width: 280px) {
2. .imageBox img {
3. min-width: 150px;
4. }
5. }

Так вот мы написали сетку, она у нас адаптивна. И теперь заказчик говорит нам, а поменяй блоки местами, к примеру блок 1 и 3 между собой на основном разрешении.

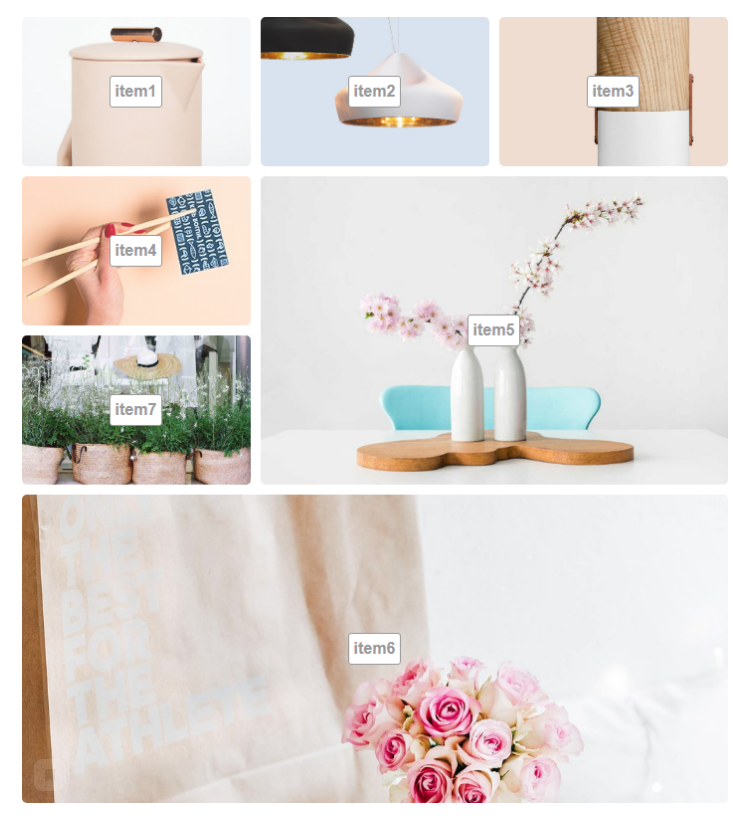
Для этого вы переходите в файл css, и меняете их местами.

1. .grid-container {
2. …
3. "box-1 box-2 box-3"
4. "box-1 box-4 box-4"
5. }

Притом, просматривая на отображение в других устройствах, их порядок остается предыдущим.

**Задание для самостоятельного выполнения**

**1. Создать галерею**



1. **Создать страницу для интернет магазина**

