**Что такое Grid Layout. Grid Container**

Grid Layout представляет специальный модуль CSS3, который позволяет позиционировать элементы в виде **сетки** или **таблицы**. Как и Flexbox, Grid Layout представляет гибкий подход к компоновке элементов, только если flexbox размещает вложенные элементы в одном направлении - по горизонтали в виде столбиков или по вертикали в виде строк, то Grid позиционирует элементы сразу в двух направлениях - в виде строк и столбцов, образуя тем самым таблицу.

**Поддержка браузерами**

При использовании Grid Layout следует учитывать, что только относительно недавно производители браузеров стали внедрять поддержку этого модуля в свои браузеры. Ниже приводится для браузеров список версий, начиная с которых была внедрена полноценная поддержка Grid Layout:

* Google Chrome - с версии 57
* Mozilla Firefox - с версии 52
* Opera - с версии 44
* Safari - с версии 10.1
* iOS Safari - с версии 10.3

Как можно заметить, большинство версий браузеров вышли в начале 2017 года. То есть на более старые версии этих браузеров рассчитывать не приходится.

Кроме того, IE (начиная с версии 10) и Microsoft Edge имеет лишь частичную поддержку модуля.

**Создание grid-контейнера**

Основой для определения компоновки Grid Layout является grid container, внутри которого размещаются элементы. Для создания grid-контейнера необходимо присвоить его стилевому свойству display одно из двух значений: **grid** или **inline-grid**.

Создадим простейшую веб-страницу, которая применяет Grid Layout:

<!DOCTYPE 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-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;}

</style>

</head>

<body>

<div class="grid-container">

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

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

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

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

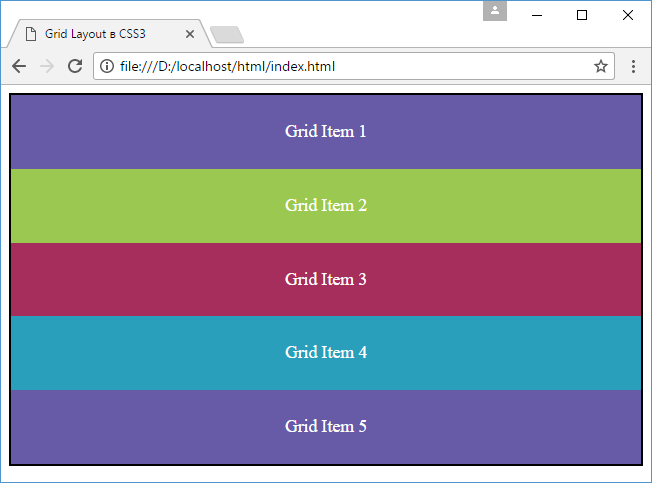
<div class="grid-item color1">Grid Item 5</div>

</div>

</body>

</html>

Для контейнера grid-container установлено свойство **display:grid**. В нем располагается пять grid-элементов.



Если значение grid определяет контейнер как блочный элемент, то значение **inline-grid** определяет элемент как строчный (inline):

<!DOCTYPE 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: inline-grid;

}

.grid-item {

box-sizing: border-box;

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;}

</style>

</head>

<body>

<div class="grid-container">

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

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

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

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

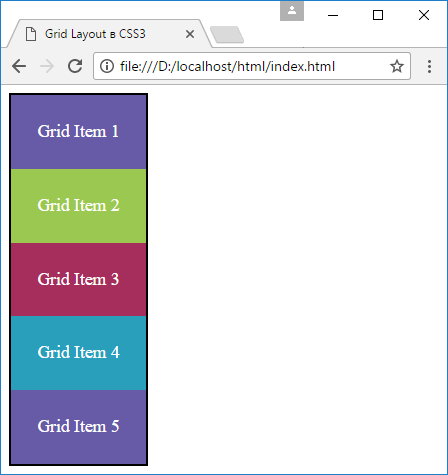
<div class="grid-item color1">Grid Item 5</div>

</div>

</body>

</html>

В этом случае весь грид занимает только то пространство, которое необходимо для размещения его элементов.



**Строки и столбцы**

Грид образует сетку из строк и столбцов, на пересечении которых образуются ячейки. И для установки строк и столбцов в Grid Layout использовать следующие свойства CSS3:

* **grid-template-columns** - настраивает столбцы;
* **grid-template-rows** - настраивает строки.

### Столбцы

Для определения столбцов используем у grid-контейнера стилевое свойство **grid-template-columns**. Например, определим грид с двумя столбцами:

<!DOCTYPE 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: 8em 8em;

}

.grid-item {

box-sizing: border-box;

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;}

</style>

</head>

<body>

<div class="grid-container">

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

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

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

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

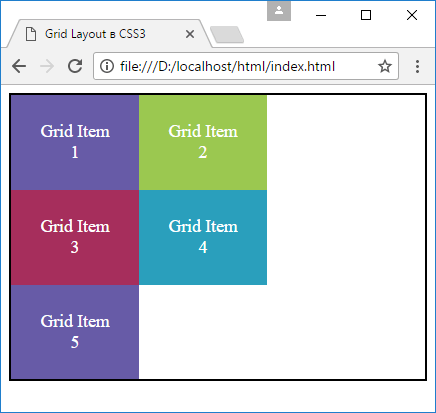
<div class="grid-item color1">Grid Item 5</div>

</div>

</body>

</html>

В качестве значения свойству **grid-template-columns** передается ширина столбцов. Сколько нужно иметь в гриде столбцов, столько и нужно передать значений этому свойству. Так, в случае выше грид содержит два столбца, поэтому свойству передаются два значения, которые указывают ширину столбцов:



Если мы хотим, чтобы в гриде было три столбца, то надо передать три значения, например

grid-template-columns: 8em 7em 8em;

Если столбцов больше чем элементов, то по умолчанию для их вмещения создаются новые строки.

### Строки

Настойка строк во многом аналогичная настройке столбцов. Для этого у грид-контейнера необходимо установить свойство **grid-template-rows**, которое задает количество и размеры строк:

<!DOCTYPE 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-rows: 4em 5em;

}

.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;}

</style>

</head>

<body>

<div class="grid-container">

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

<div class="grid-item color2">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 color1">Grid Item 5</div>

<div class="grid-item color4">Grid Item 6</div>

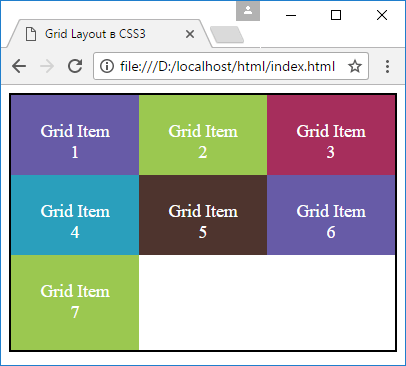
<div class="grid-item color3">Grid Item 7</div>

</div>

</body>

</html>

Свойству **grid-template-rows** передается высота каждой из строк. Так, в данном случае высота первой строки составляет 4em, а второй - 5em.



Если элементов больше, чем ячеек грида, то образуются дополнительные строки (как в случае со столбцами). Поэтому, несмотря на то, что выше были определены настройки только для двух строк, в реальности строк в гриде будет три строки, причем, как видно на скриншоте, высота третьей строки необязательно будет 5em, как у других строк, она будет вычисляться автоматически.

Изменим стили grid-контейнера, добавив еще две строки:

.grid-container {

border: solid 2px #000;

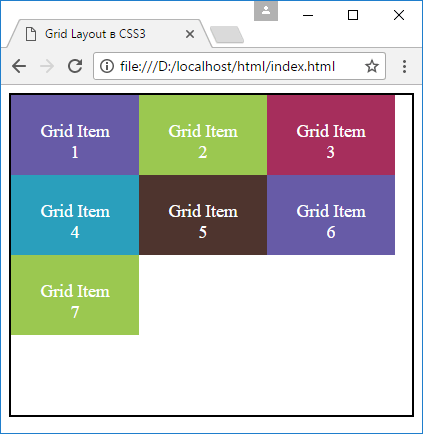
display: grid;

grid-template-columns: 8em 8em 8em;

grid-template-rows: 5em 5em 5em 5em;

}

Здесь определено три столбца и четыре строки:



Поскольку ячеек грида больше, чем элементов, то последняя строка оказывается пустой.

## **Функция repeat и свойство grid**

**Повторение строк и столбцов**

Если столбцов и(или) строк много и они имеют одинаковые размеры, то есть смыл использовать специальную функцию **repeat( )**, которая позволит настроить строки и столбцы. Так, в примере выше повторяется определение одинаковых строк и столбцов в grid-контейнере:

grid-template-columns: 8em 8em 8em;

grid-template-rows: 5em 5em 5em 5em;

Происходит повторение одних и тех же размеров - 8em и 5em для установки ширины столбцов и высоты строк. Перепишем стили, применив функцию repeat:

grid-container {

border: solid 2px #000;

display: grid;

grid-template-columns: repeat(3, 8em);

grid-template-rows: repeat(4, 5em);

}

Первый параметр функции **repeat** представляет число повторений, а второй - определение строк или столбцов. Например, свойство grid-template-columns: repeat(3, 8em); говорит, что необходимо определить 3 столбца шириной в 8em.

Соответственно выражение grid-template-rows: repeat(4, 5em) определяет 4 строки высотой по 5em.

Можно задавать повторение нескольких столбцов и строк:

.grid-container {

border: solid 2px #000;

display: grid;

grid-template-columns: repeat(2, 7em 8em);

grid-template-rows: 6em repeat(3, 5em);

}

В данном случае будет создано 4 столбца: два раза будут повторяться два столбца с шириной 7em и 8em. В случае со строками будет создано 4 строки. Причем первая будет иметь высоту в 6em, а остальные три - 5em.

**Свойство grid**

Свойство **grid** объединяет свойства grid-template-rows и grid-template-columns и разом позволяет задать настройки для строк и столбцов в следующем формате:

grid: grid-template-rows / grid-template-columns;

К примеру, есть следующее определение класса grid-контейнера:

.grid-container {

border: solid 2px #000;

display: grid;

grid-template-columns: 8em 8em 8em;

grid-template-rows: 5em 5em 5em 5em;

}

Можно сократить этот класс следующим образом:

.grid-container {

border: solid 2px #000;

display: grid;

grid: 5em 5em 5em 5em / 8em 8em 8em;

}

Используя функцию **repeat( )**, можно еще больше сократить определение грида:

.grid-container {

border: solid 2px #000;

display: grid;

grid: repeat(4, 5em) / repeat(3, 8em);

}

## **Размеры строк и столбцов**

**Фиксированные размеры**

Для определения размеров можно использовать самые различные единицы измерения, которые доступны в CSS (px, em, rem, pt, %), например:

.grid-container {

border: solid 2px #000;

display: grid;

grid-template-columns: repeat(3, 200px);

grid-template-rows: repeat(3, 4.5em);

}

**Автоматические размеры**

Кроме точных размеров можно задавать автоматические размеры с помощью слова **auto**. В этом случае ширина столбцов и высота строк вычисляются исходя из размеров содержимого:

<!DOCTYPE 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: 8em auto auto;

grid-template-rows: auto 4.5em auto;

}

.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">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>

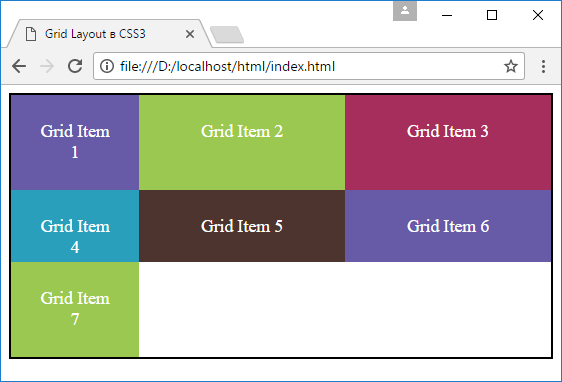
<div class="grid-item color2">Grid Item 7</div>

</div>

</body>

</html>

Задано три строки и три столбца. Первый столбец имеет фиксированную ширину в 8em, а второй и третий столбцы - автоматическую ширину. И также первая и третья строки имеют автоматическую высоту, а вторая строка - фиксированную.



**Пропорциональные размеры**

Для установки пропорциональных размеров применяется специальная единица измерения **fr**. Она представляет собой часть пространства (fraction), которое отводится для данного столбца или строки. Значение fr еще называют flex-фактором (flex factor).

Вычисление пропорциональных размеров производится по формуле:

flex-фактор \* доступное\_пространство / сумма всех flex-факторов

При этом под доступным пространством понимается все пространство grid-контейнера за исключением фиксированных значений строк и столбцов.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

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

<title>Grid Layout в CSS3</title>

<style>

\*{

box-sizing: border-box;

}

html, body{

margin:0;

padding:0;

}

.grid-container {

height: 100vh;

border: solid 2px #000;

display: grid;

grid-template-columns: 8em 2fr 1fr;

grid-template-rows: 1fr 4.5em 1fr;

}

.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">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 color2">Grid Item 7</div>

</div>

</body>

</html>

В данном случае имеются три столбца с шириной 2fr, 8em, 1fr. Поэтому ширина второго столбца будет вычисляться по формуле:

2 \* (ширина\_грида - 8em) / (2 + 1)

Ширина третьего столбца будет вычисляться по формуле:

1 \* (ширина\_грида - 8em) / (2 + 1)

И если первый столбец фиксированный с шириной 8em, то ширина второго и третьего столбца будут зависеть от ширины контейнера и будут автоматически масштабироваться при ее изменении. В отношении строк все аналогично.

## **Отступы между столбцами и строками**

Для создания отступов между столбцами и строками применяются свойства **grid-column-gap** и **grid-row-gap** соответственно.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

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

<title>Grid Layout в CSS3</title>

<style>

\*{

box-sizing: border-box;

}

html, body{

margin:0;

padding:0;

}

.grid-container {

height: 100vh;

display: grid;

grid-template-columns: repeat(3, 1fr);

grid-template-rows: repeat(3, 1fr);

grid-column-gap: 10px;

grid-row-gap: 10px;

}

.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">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 color1">Grid Item 5</div>

<div class="grid-item color4">Grid Item 6</div>

<div class="grid-item color5">Grid Item 7</div>

</div>

</body>

</html>



Если значения свойств **grid-column-gap** и **grid-row-gap** совпадают, то вместо них можно определить одно свойство **grid-gap**, которое установит оба отступа:

.grid-container {

height: 100vh;

display: grid;

grid-template-columns: repeat(3, 1fr);

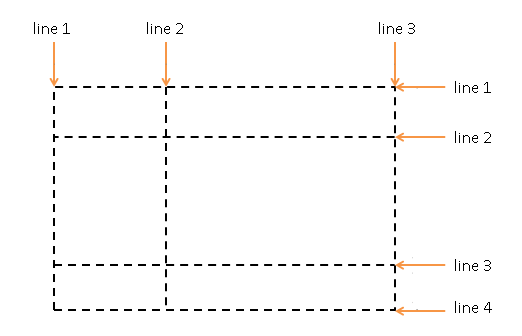
grid-template-rows: repeat(3, 1fr);

grid-gap: 10px;

}

## **Позиционирование элементов**

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 можно 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:

<!DOCTYPE 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>

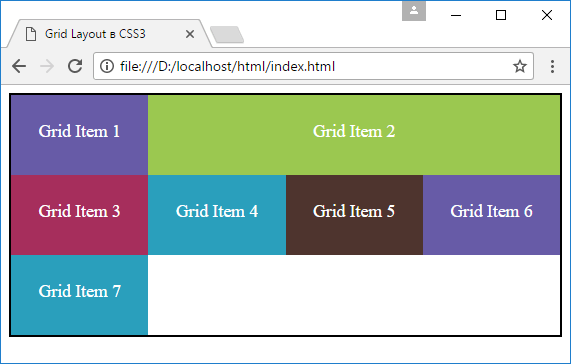
<div class="grid-item color1">Grid Item 6</div>

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

</div>

</body>

</html>



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).

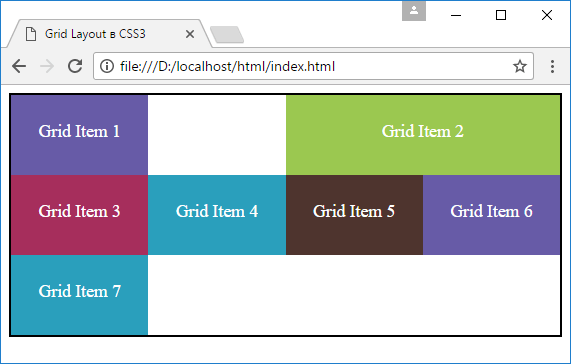
Второй элемент необязательно должен начинаться со второго столбца, это может быть любой другой: и 1-й, и 3-й и т.д. Например, если мы поместим второй элемент, начиная с 3-го столбца, то на месте второго столбца образуется пустота:

.special-item{

grid-column-start: 3;

grid-column-end: 5;

}



Если для второго элемента установить начало с первого столбца, то второй элемент будет перенесен на следующую строку, и таким образом, он будет начинаться с первого столбца.

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;

Можно 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-item следующим образом:

.special-item{

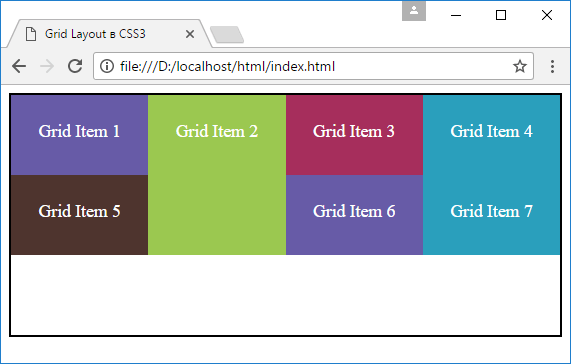
grid-column-start:2;

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;

Можно изменить стиль special-item следующим образом:

.special-item{

grid-column-start:2;

grid-row: 1 / 3;

}

### span

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

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;

}

## **Наложение элементов**

Манипулируя положением элементов, можем осуществить их наложение, создать своего рода слои из элементов. Например, определим следующий грид:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

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

<title>Grid Layout в CSS3</title>

<style>

\*{

box-sizing: border-box;

}

html, body{

margin:0;

padding:0;

}

.grid-container {

height: 100vh;

display: grid;

grid-template-columns: repeat(3, 1fr);

grid-template-rows: repeat(3, 1fr);

grid-gap: 10px;

}

.grid-item {

text-align:center;

font-size: 1.1em;

padding: 1.5em;

color: white;

z-index: 1;

}

.item1{

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

opacity: 0.8;

}

.item2{

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

}

.item3{

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

}

.item4{

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

}

.item5{

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

}

.item6{

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

}

.item7{

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

}

.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 item1 color2">Grid Item 2</div>

<div class="grid-item item2 color1">Grid Item 1</div>

<div class="grid-item item3 color3">Grid Item 3</div>

<div class="grid-item item4 color4">Grid Item 4</div>

<div class="grid-item item5 color1">Grid Item 5</div>

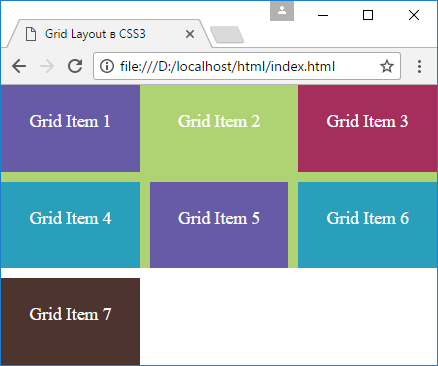
<div class="grid-item item6 color4">Grid Item 6</div>

<div class="grid-item item7 color5">Grid Item 7</div>

</div>

</body>

</html>



Здесь первый элемент занимает то же пространство, что и пять следующих элементов. Благодаря установке свойства z-index: 1 у всех элементов получается, что как первый элемент как-будто лежит фоном под другими пятью элементами.

При необходимости слой из первого элемента можно наоборот перенести ближе к пользователю, накрыв остальные элементы. Для этого надо увеличить у него значение z-index:

.item1{

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

opacity: 0.8;

z-index: 10;

}

## 

## **Направление и порядок элементов**

### Свойство grid-auto-flow

По умолчанию все элементы располагаются по порядку горизонтально, если места в строке больше нет, то элементы переносятся на следующую строку. Но с помощью свойства **grid-auto-flow** можно изменить направление элементов. Это свойство принимает два значения:

* **row**: значение по умолчанию, элементы располагаются в строку друг за другом, если места в строке не хватает, элементы переносятся на следующую строку
* **column**: элементы располагаются в столбик, если места в столбце не хватает, то элементы переходят в следующий столбец

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

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

<title>Grid Layout в CSS3</title>

<style>

\*{

box-sizing: border-box;

}

html, body{

margin:0;

padding:0;

}

.grid-container {

height: 100vh;

display: grid;

grid-template-columns: repeat(3, 1fr);

grid-template-rows: repeat(3, 1fr);

grid-auto-flow: row;

}

.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">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 color1">Grid Item 5</div>

<div class="grid-item color4">Grid Item 6</div>

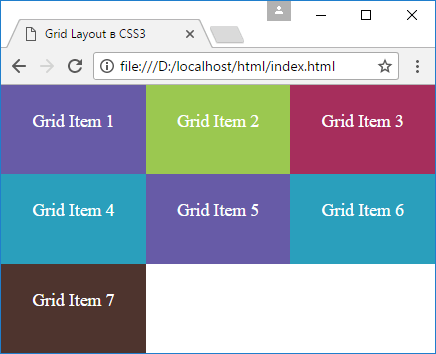
<div class="grid-item color5">Grid Item 7</div>

</div>

</body>

</html>

Свойство grid-auto-flow имеет значение row, поэтому элементы будут располагаться в строку.



Теперь изменим стиль grid-контейнера:

.grid-container {

height: 100vh;

display: grid;

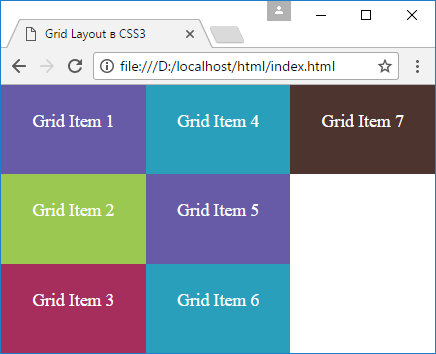
grid-template-columns: repeat(3, 1fr);

grid-template-rows: repeat(3, 1fr);

grid-auto-flow: column;

}

После этого элементы будут располагаться в столбец



### Свойство order

Свойство **order** позволяет задать порядок элементов. По умолчанию для каждого элемента в гриде это свойство имеет значение 0. Поэтому элементы располагаются друг за другом как они определены в разметке html. Но можно переопределить этот порядок:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

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

<title>Grid Layout в CSS3</title>

<style>

\*{

box-sizing: border-box;

}

html, body{

margin:0;

padding:0;

}

.grid-container {

height: 100vh;

display: grid;

grid-template-columns: repeat(3, 1fr);

grid-template-rows: repeat(3, 1fr);

}

.grid-item {

text-align:center;

font-size: 1.1em;

padding: 1.5em;

color: white;

}

.last-item{

order: 1;

}

.first-item{

order: -1;

}

.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 last-item">Grid Item 2</div>

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

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

<div class="grid-item color1">Grid Item 5</div>

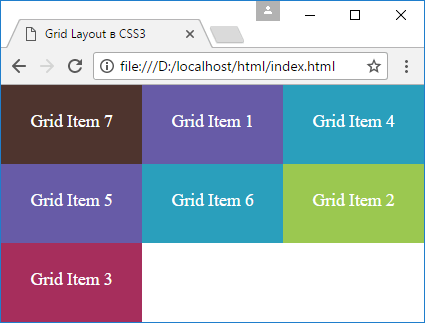
<div class="grid-item color4">Grid Item 6</div>

<div class="grid-item color5 first-item">Grid Item 7</div>

</div>

</body>

</html>



Элементы с классом last-item имеет порядок 1, поэтому они будут располагаться после других элементов, у которых порядок равен 0 или меньше.

Если же необходимо поставить одни элементы перед другими, то можно использовать отрицательное значение для свойства order. Так, у седьмого элемента порядок равен -1, что меньше, чем у других элементов. Поэтому последний в разметке элемент по факту будет располагаться перед всеми элементами.

## **Именованные grid-линии**

В Grid Layout мы можем дать наименование каждой линии грида, присвоив ей какое-либо имя в квадратных скобках и затем, используя это имя, позиционировать элементы. Например, определим следующую веб-страницу:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

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

<title>Grid Layout в CSS3</title>

<style>

\*{

box-sizing: border-box;

}

html, body{

margin:0;

padding:0;

}

.grid-container {

height:100vh;

display: grid;

grid-template-columns: [col1start] 1fr [col1end] 10px

[col2start] 1fr [col2end] 10px

[col3start] 1fr [col3end];

grid-template-rows: [row1start] 1fr [row1end] 10px [row2start] 1fr [row2end];

}

.grid-item {

background-color: #ddd;

}

.special-item{

grid-column: col1start / col2end;

grid-row: row1start;

background-color: #bbb;

}

.item1{

grid-column: col3start / col3end;

grid-row: row1start;

}

.item2{

grid-column: col1start / col1end;

grid-row: row2start;

}

.item3{

grid-column: col2start / col2end;

grid-row: row2start;

}

.item4{

grid-column: col3start / col3end;

grid-row: row2start;

}

</style>

</head>

<body>

<div class="grid-container">

<div class="grid-item special-item"></div>

<div class="grid-item item1"></div>

<div class="grid-item item2"></div>

<div class="grid-item item3"></div>

<div class="grid-item item4"></div>

</div>

</body>

</html>

При именовании линий их имена заключаются в квадратные скобки, а между для именами указывается ширина столбца или высота строки, которые находятся между этими линиями:

grid-template-columns: [col1start] 1fr [col1end] 10px

[col2start] 1fr [col2end] 10px

[col3start] 1fr [col3end];

grid-template-rows: [row1start] 1fr [row1end] 10px [row2start] 1fr [row2end];

Затем, используя эти названия, мы можем позиционировать элементы между определенными линиями:

.special-item{

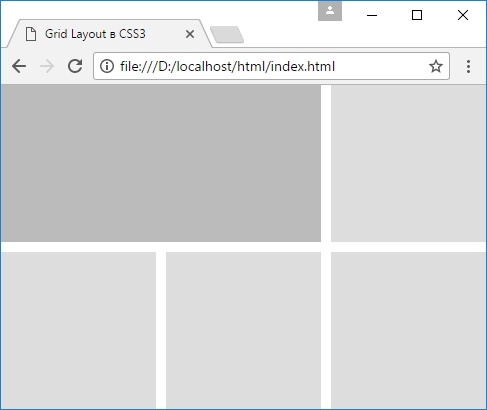
grid-column: col1start / col2end;

grid-row: row1start;

background-color: #bbb;

}

Так, в данном случае элемент с классом special-item начинает от вертикальной линии col1start и растягивается до вертикальной линии col2end. И также он начинается от горизонтальной линии row1start, и так как конечная горизонтальная линия не указана, то элемент занимает только одну строку.



Или другой пример, связанный с созданием макета страницы:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

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

<title>Grid Layout в CSS3</title>

<style>

\*{

box-sizing: border-box;

}

html, body {

padding: 0;

margin: 0;

}

.grid-container {

display: grid;

height: 100vh;

grid-template-rows: 6em 10px 1fr;

grid-template-columns: [mainstart] 1fr [mainend] 10px [sidebarestart] 25% [sidebarend];

}

.grid-item-content {

background-color: #455a64;

grid-row: 3 / 4;

grid-column: mainstart / mainend;

}

.grid-item-sidebar {

background-color: #37474f;

grid-row: 3 / 4;

grid-column: sidebarestart / sidebarend;

}

.grid-item-header{

background-color: #263238;

grid-row: 1 / 2;

grid-column: mainstart / sidebarend;

}

.grid-item {

color: #fff;

text-align: center;

}

</style>

</head>

<body>

<main class="grid-container">

<header class="grid-item grid-item-header">

<h1>Header</h1>

</header>

<article class="grid-item grid-item-content">

<h1>Main Content</h1>

</article>

<aside class="grid-item grid-item-sidebar">

<h1>Sidebar</h1>

</aside>

</main>

</body>

</html>

## 

## **Именованные grid-линии и функция repeat**

С помощью ранее рассмотренной функции repeat мы можем растиражировать столбцы и строки, которые создаются между именованными grid-линиями:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

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

<title>Grid Layout в CSS3</title>

<style>

\*{

box-sizing: border-box;

}

html, body{

margin:0;

padding:0;

}

.grid-container {

height:100vh;

display: grid;

grid-template-columns: 10px repeat(3, [column] 1fr [colgutter] 10px);

grid-template-rows: 10px repeat(2, [row] 1fr [rowgutter] 10px);

}

.grid-item {

background-color: #ddd;

}

.special-item{

grid-column: column 2;

grid-row: row 1;

background-color: #bbb;

}

.item1{

grid-column: column 1;

grid-row: row 1;

}

.item2{

grid-column: column 3;

grid-row: row 1;

}

.item3{

grid-column: column 1;

grid-row: row 2;

}

.item4{

grid-column: column 2;

grid-row: row 2;

}

</style>

</head>

<body>

<div class="grid-container">

<div class="grid-item special-item"></div>

<div class="grid-item item1"></div>

<div class="grid-item item2"></div>

<div class="grid-item item3"></div>

<div class="grid-item item4"></div>

</div>

</body>

</html>

Возьмем из данного примера определение столбцов:

grid-template-columns: 10px repeat(3, [column] 1fr [colgutter] 10px);

Первый столбец будет иметь ширину в 10 пикселей. Затем происходит тиражирование столбцов с помощью функции repeat. Она создает подряд три копии двух столбцов. Первый столбец имеет ширину 1fr, то есть имеет пропорциональные размеры, и располагается между grid-линиями "column" и "colgutter". После grid-линии "colgutter" идет еще один столбец шириной в 10 пикселей. И эти два столбца будут повторяться три раза. То есть всего в гриде будет 7 столбцов.

Со строками будет во многом аналогично, только там создается 5 строк с помощью grid-линий "row" и "rowgutter".

При определении стиля элементов, используя имя grid-линий и их порядковый номер, мы можем явным образом указать с помощью свойств**grid-column** и **grid-row**, где именно должен располагаться элемент:

.special-item{

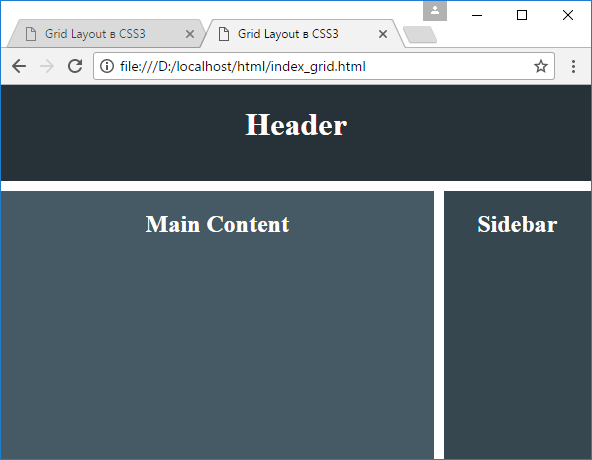
grid-column: column 2; /\* второй столбец с именем column \*/

grid-row: row 1; /\* первая строка с именем row \*/

background-color: #bbb;

}

Причем свойство grid-column: column 2 указывает на столбец в гриде, который начинается со второй grid-линии "column". В итоге мы получим следующий грид:



Возможно, многие найдут такой подход более интуитивно понятным для определения позиции элемента.

И более того мы можем дополнительно добавлять новые именованные грид-линии вне функции repeat:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

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

<title>Grid Layout в CSS3</title>

<style>

\*{

box-sizing: border-box;

}

html, body{

margin:0;

padding:0;

}

.grid-container {

height:100vh;

display: grid;

grid-template-columns: 10px repeat(3, [column] 1fr [colgutter] 10px)

[sidebarstart] 150px [sidebarend] 10px;

grid-template-rows: 10px repeat(2, [row] 1fr [rowgutter] 10px);

}

.grid-item {

background-color: #ddd;

}

.special-item{

grid-column: column 2;

grid-row: row 1;

background-color: #bbb;

}

.item1{

grid-column: column 1;

grid-row: row 1;

}

.item2{

grid-column: column 3;

grid-row: row 1;

}

.item3{

grid-column: column 1;

grid-row: row 2;

}

.item4{

grid-column: column 2;

grid-row: row 2;

}

.sidebar{

grid-column: sidebarstart / sidebarend;

grid-row: 2 / 5;

background-color: #ccc;

}

</style>

</head>

<body>

<div class="grid-container">

<div class="grid-item special-item"></div>

<div class="grid-item item1"></div>

<div class="grid-item item2"></div>

<div class="grid-item item3"></div>

<div class="grid-item item4"></div>

<div class="grid-item sidebar"></div>

</div>

</body>

</html>

Стоит отметить, что вне зависимости от того, именованные строки,столбцы и grid-линии или неименованные, мы по прежнему можем позиционировать элементы, используя номера grid-линий, как в данном случае происходит в отношении сайдбара:

.sidebar{

grid-column: sidebarstart / sidebarend;

grid-row: 2 / 5;

background-color: #ccc;

}

## 

## **Области грида**

В рамках грида можем определять области (grid area). Области определятся с помощью двух вертикальных и двух горизонтальных grid-линий, которые собственно и задают занимаемое областью пространство. В этом плане область не эквивалентна одной ячейке грида и может включать несколько ячеек. Области особенно полезны для определения семантических отношений между различными частями макета страницы.

Для определения областей у grid-контейнера применяется свойство **grid-template-areas**. Например, определим три области:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

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

<title>Grid Layout в CSS3</title>

<style>

\*{

box-sizing: border-box;

}

html, body{

margin:0;

padding:0;

}

.grid-container {

height:100vh;

display: grid;

grid-template-areas: "header header"

"sidebar content"

"sidebar content";

grid-template-columns: 150px 1fr;

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

}

.header { grid-area: header; background-color: #bbb; }

.sidebar { grid-area: sidebar; background-color: #ccc; }

.content { grid-area: content; background-color: #eee; }

</style>

</head>

<body>

<div class="grid-container">

<div class="header"></div>

<div class="sidebar"></div>

<div class="content"></div>

</div>

</body>

</html>

Здесь у grid-контейнера определяется два столбца и три строки:

grid-template-columns: 150px 1fr;

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

То есть в итоге у нас будет в гриде 3 х 2 = 6 ячеек. Но в разметке страницы определено три элемента с одноименными областями: header, sidebar, content. И свойство grid-template-areas как раз устанавливает, как эти области будут располагаться в ячейках грида:

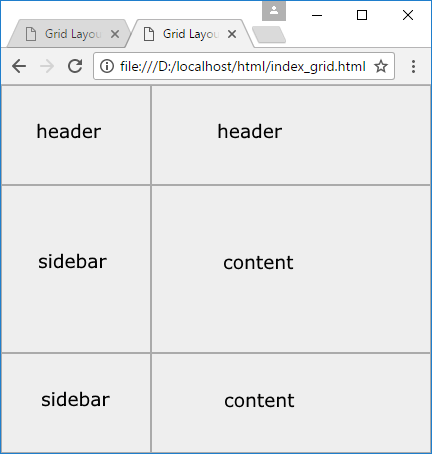
grid-template-areas: "header header"

"sidebar content"

"sidebar content";

Выражение "header header" представляет первую строку и указывает, что область header занимает две ячейки подряд.

Следующее выражение "sidebar content" указывает на вторую строку, где область sidebar занимает первую ячейку, а content - вторую ячейку. Третья строка повторяет вторую. То есть в итоге, если смотреть по ячейкам, то получится следующим образом:



Для установки области у элементов задается свойство **grid-area**:

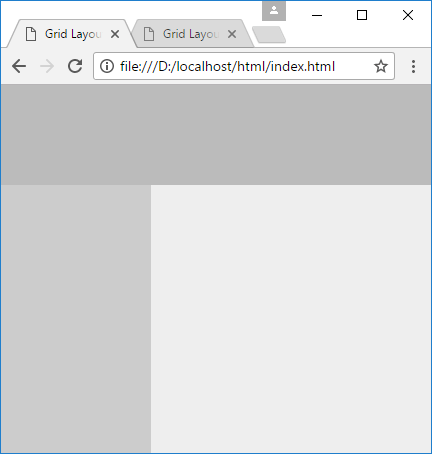
.header {

grid-area: header; /\* элемент с классом header помещается в область header\*/

background-color: #bbb;

}

В итоге получим следующую страницу:



Рассмотрим другой более сложный пример. Пусть код веб-страницы выглядит следующим образом:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

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

<title>Grid Layout в CSS3</title>

<style>

\*{

box-sizing: border-box;

}

html, body{

margin:0;

padding:0;

}

.grid-container {

height:100vh;

display: grid;

grid-template-areas: "header header header header header"

". . . . ."

"menu . content . sidebar"

". . . . ."

"footer footer footer footer footer";

grid-template-columns: 130px 5px 1fr 5px 130px;

grid-template-rows: 90px 5px 1fr 5px 90px;

}

.header { grid-area: header; background-color: #bbb; }

.menu { grid-area: menu; background-color: #ccc; }

.sidebar { grid-area: sidebar; background-color: #ccc; }

.content { grid-area: content; background-color: #eee; }

.footer { grid-area: footer; background-color: #bbb; }

</style>

</head>

<body>

<div class="grid-container">

<div class="header"></div>

<div class="content"></div>

<div class="menu"></div>

<div class="sidebar"></div>

<div class="footer"></div>

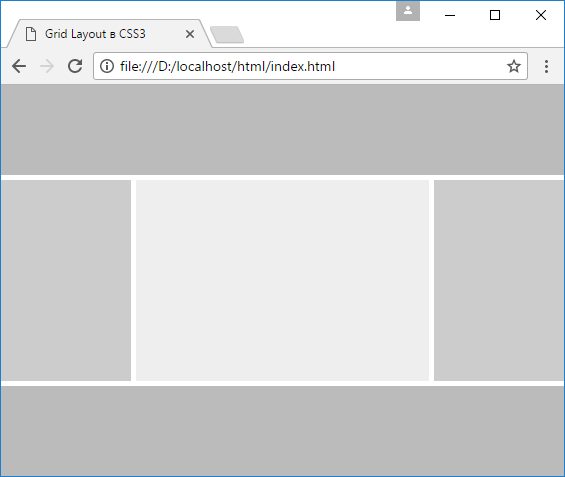
</div>

</body>

</html>

Теперь грид содержит 5 строк и 5 столбцов, то есть в совокупности 25 ячеек, которые размещают 5 областей: header, menu, sidebar, content и footer.

Свойство **grid-template-areas** в стилях grid-контейнера опять же содержит определение того, как все эти области сопоставляются с ячейками. Например, элемент header должен занимать все пять ячеек первой строки: "header header header header header". Но определение второй строки, высота которой 5px, выглядит уже необычно: ". . . . .". Точка означает, что данная ячейка не будет принадлежать ни одной области и останется незаполненной. Если надо оставить 5 незаполненных ячеек, то указывается пять точек, между которыми ставятся пробелы. В итоге мы получим пять областей, между которыми будут располагаться незаполненные пространства:



## **Макет страницы в Grid Layout**

Рассмотрим создание простейшего адаптивного стандартного макета веб-страницы, который состоит из шапки, подвала, основного содержимого, блока навигации и сайдбара. Для этого определим следующую веб-станицу:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

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

<title>Grid Layout в CSS3</title>

<style>

\*{

box-sizing: border-box;

}

html, body{

margin:0;

padding:0;

}

.grid-container {

height:100vh;

display: grid;

grid-template-areas: "header"

"."

"menu"

"."

"content"

"."

"sidebar"

"."

"footer";

grid-template-columns: 1fr;

grid-template-rows: 80px 5px 80px 5px 1fr 5px 80px 5px 80px;

}

.header { grid-area: header; background-color: #bbb; }

.menu { grid-area: menu; background-color: #ccc; }

.sidebar { grid-area: sidebar; background-color: #ccc; }

.content { grid-area: content; background-color: #eee; }

.footer { grid-area: footer; background-color: #bbb; }

h3{

text-align:center;

}

@media screen and (min-width: 468px) {

.grid-container {

height:100vh;

display: grid;

grid-template-areas: "header header header header header"

". . . . ."

"menu . content . sidebar"

". . . . ."

"footer footer footer footer footer";

grid-template-columns: 130px 5px 1fr 5px 130px;

grid-template-rows: 90px 5px 1fr 5px 90px;

}

}

</style>

</head>

<body>

<div class="grid-container">

<div class="header"><h3>Header</h3></div>

<div class="content"><h3>Content</h3></div>

<div class="menu"><h3>Menu</h3></div>

<div class="sidebar"><h3>Sidebar</h3></div>

<div class="footer"><h3>Footer</h3></div>

</div>

</body>

</html>

В прошлой теме уже рассматривалось применение именованных областей. И здесь они также используются, так как очень удобно связывать стили с семантикой станицы через области.

Однако, так как предполагается, что веб-станицу могут просматривать на мобильных устройствах, где ширина экрана ограничена, то хотелось бы иметь адаптивный макет. Поэтому в стиля имеются два разных определения грида. Одно определение грида для мобильных устройств (условно в качестве ширины устройств выбрано значение в 468px):

.grid-container {

height:100vh;

display: grid;

grid-template-areas: "header"

"."

"menu"

"."

"content"

"."

"sidebar"

"."

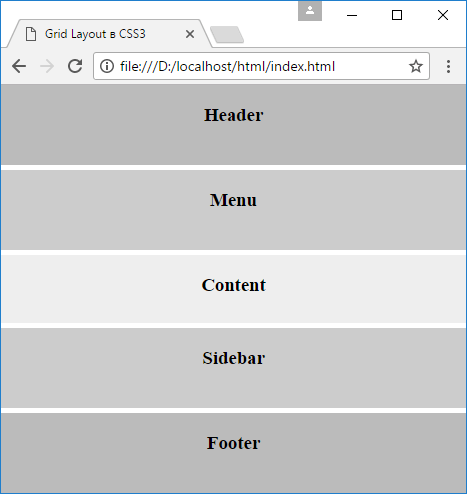
"footer";

grid-template-columns: 1fr;

grid-template-rows: 80px 5px 80px 5px 1fr 5px 80px 5px 80px;

}

В таком состоянии грид имеет только один столбец и 5 строк для каждой области плюс 4 строки-разделители.



При увеличении ширины экрана в действие вступает другое определение грида:

.grid-container {

height:100vh;

display: grid;

grid-template-areas: "header header header header header"

". . . . ."

"menu . content . sidebar"

". . . . ."

"footer footer footer footer footer";

grid-template-columns: 130px 5px 1fr 5px 130px;

grid-template-rows: 90px 5px 1fr 5px 90px;

}

В этом случае столбцов и строк будет по пять:

