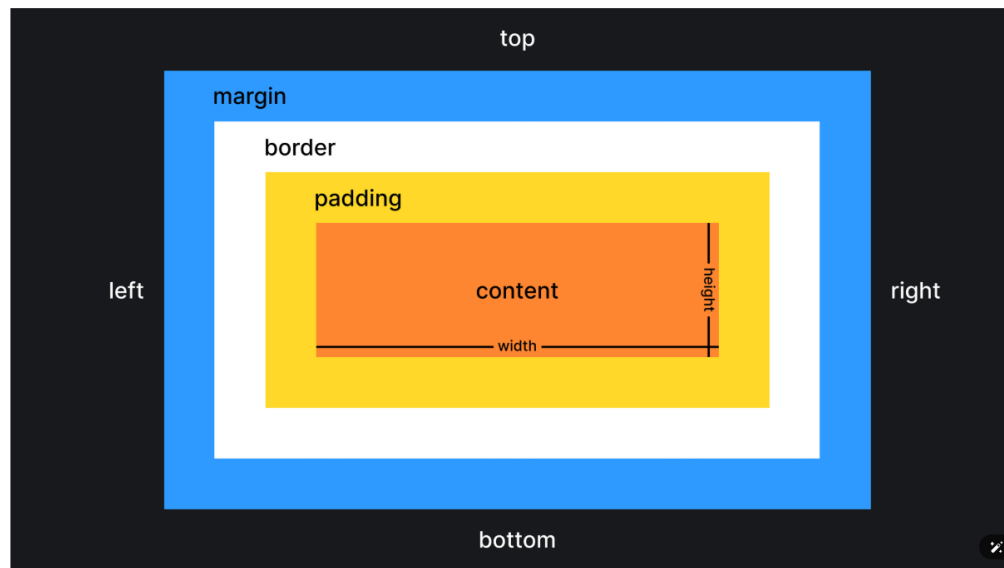


Лабораторная работа 6. Блочная модель

Браузеры рисуют любой элемент на HTML-странице как прямоугольник. Как рассчитывается размер этого прямоугольника? Разберёмся с одной из основных концепций вёрстки.

Блочная модель, она же box model — это алгоритм расчёта размеров каждого отдельного элемента на странице, которым браузеры пользуются при отрисовке. Чтобы точно понимать, каким в итоге получится блок и сколько места он займёт, посмотрите на рисунок:



Блочная модель состоит из нескольких CSS-свойств, влияющих на размеры элемента:

- width — ширина элемента;
- height — высота элемента;

- padding — внутренние отступы от контента до краёв элемента;
- border — рамка, идущая по краю элемента;
- margin — внешние отступы вокруг элемента.

Такую же схему, но в других цветах можно увидеть в инструментах разработчика любого из браузеров.

Например, так выглядит блочная модель элемента в Chrome.

Создайте файлы index.html и style.css

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=0.5">
6          <title>Изучаем CSS</title>
7          <link rel="stylesheet" href="css/style.css">
8      </head>
9      <body>
10         <div class="example"></div>
11     </body>
12 </html>
```

```
1  .example {  
2    background-color: navy;  
3    width: 400px;  
4    padding: 20px;  
5    border: 6px solid gray;  
6    /* box-sizing: border-box; */  
7  }
```

Результат:



Откройте DevTools, нажав F12, и выделите интересующий нас блок:

Элементы

Консоль

Источники

Сеть

Производительность

Память

Приложение

Безопасность

>>

<!DOCTYPE html>

<html>

▶<head>⋮</head>

▼<body>

...<div class="example"></div> == \$0

</body>

</html>

Стили

Вычисленные

Макет

Прослушватели событий

Точки останова DOM

Свойства

>>

Фильтр

:hov .cls

+

⌵

⌶

▲

element.style {

}

.example { style.css:1

background-color: navy;

width: 400px;

padding: ▶ 20px;

border: ▶ 6px solid gray;

☐ box-sizing: border-box;

}

таблица стилей агента пользователя

div {

display: block;

unicode-bidi: isolate;

}

margin

border

padding

400x0

Фильтр

☐ Показать все

☐ Группа

▶background-color

▶border-bottom-color

▶border-bottom-style

▶border-bottom-width

▶border-image-outset

▶border-image-repeat

▶border-image-slice

▶border-image-source

▶border-image-width

▶border-left-color

▶border-left-style

▶border-left-width

▶border-right-color

▶border-right-style

▶border-right-width

▶border-top-color

▶border-top-style

▶border-top-width

box-sizing

▶display

height

▶padding-bottom

▶padding-left

▶padding-right

▶padding-top

▶unicode-bidi

▶width

■ rgb(0, 0, 128)

■ rgb(128, 128, 128)

solid

5.6px

0

stretch

100%

none

1

■ rgb(128, 128, 128)

solid

5.6px

■ rgb(128, 128, 128)

solid

5.6px

■ rgb(128, 128, 128)

solid

5.6px

content-box

block

0px

20px

20px

20px

20px

isolate

400px

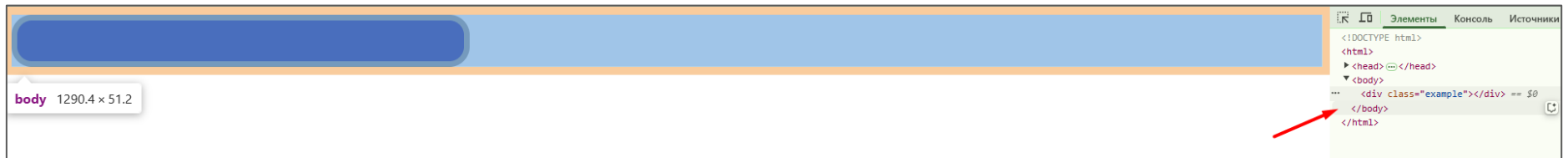
Можно явно увидеть все свойства блока. Наводите курсор мыши на рисунок справа, и слева будет подсвечиваться выделяемый элемент.

Далее наведите курсор на тег `<body>` и вы увидите, что блок занимает всю ширину страницы.



Теперь добавьте скругление элемента

Снова наведите на `<body>` и заметьте, что, не смотря на то, что появилось скругление, блок остался прямоугольным размером на ширину страницы.



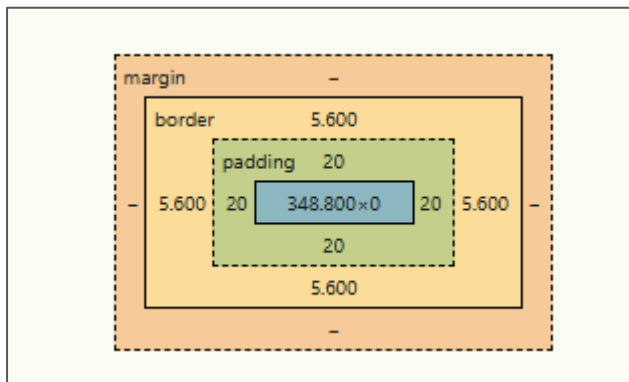
Внесите следующее изменение в файл css: уберите радиус скругления и добавьте свойство `border-box`.

```

1  .example {
2      background-color: navy;
3      width: 400px;
4      padding: 20px;
5      border: 6px solid gray;
6      box-sizing: border-box;
7  }

```

Обратите внимание, что блок уменьшился:



В чем причина? По умолчанию браузеры рассчитывают размеры элемента ровно так, как описано выше, прибавляя внутренние отступы и рамки к ширине и высоте. С этим могут быть связаны неприятные сюрпризы, когда элемент в вёрстке занимает больше места, чем вы ожидаете.

Мы можем изменить стандартное поведение и указать браузеру, что ширина и высота, заданные в CSS, должны включать в себя, в том числе, внутренние отступы и рамки. Делается это при помощи свойства `box-sizing`.

Свойство display и верстка по сетке

Свойство display - это CSS-свойство, позволяющее настраивать способ отображения и поведения элемента относительно других элементов.

Рассмотрим четыре разных значения для свойства display, с помощью которых можно менять расположение элементов.

1. display: block.

Значение **block** для свойства display задаст элементу следующие настройки:

- элемент будет видимым на странице;
- границы элемента описывает прямоугольник;
- элемент займёт всю ширину своего родительского блока (это можно изменить с помощью свойства width);
- следующий соседний элемент будет расположен **под текущим**.

`display: block;`

Элемент 1

Элемент 2

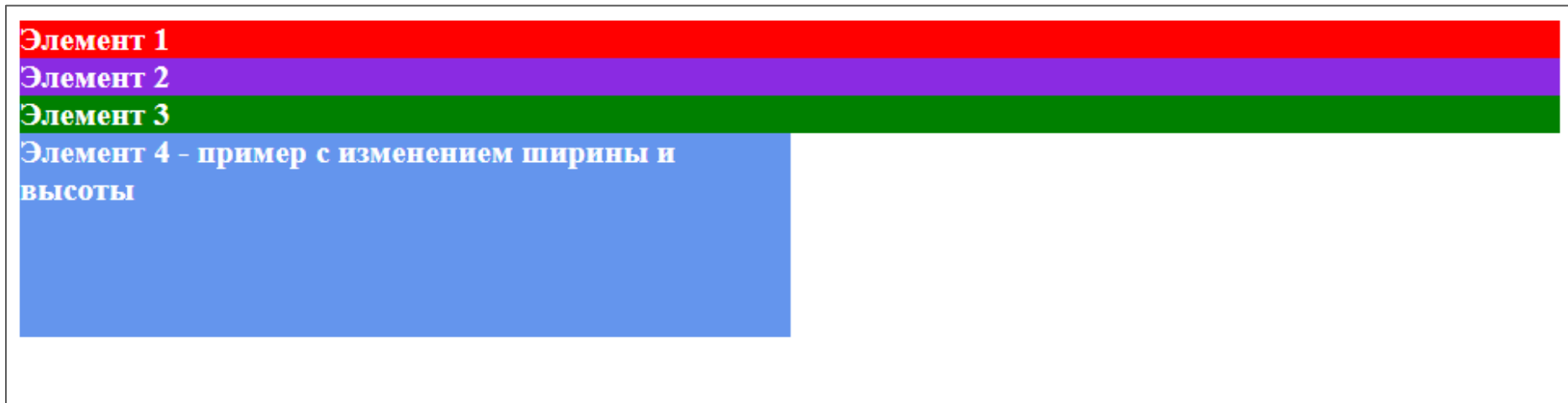
Элемент 3

Код для примера:


```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=0.5">
6          <title>Изучаем CSS</title>
7          <link rel="stylesheet" href="css/style.css">
8      </head>
9      <body>
10         <strong class="element-1">Элемент 1</strong>
11         <strong class="element-2">Элемент 2</strong>
12         <strong class="element-3">Элемент 3</strong>
13         <strong class="element-4">Элемент 4 - пример с изменением ширины и высоты</strong>
14     </body>
15 </html>
```

```
1  .element-1 {
2      display: block;
3      background-color: red; /* Цвет фона */
4      color: white; /* Цвет текста */
5  }
6
7  .element-2 {
8      display: block;
9      background-color: blueviolet; /* Цвет фона */
10     color: white; /* Цвет текста */
11 }
12
13 .element-3 {
14     display: block;
15     color: white; /* Цвет текста */
16     background-color: green; /* Цвет фона */
17 }
18
19 .element-4 {
20     display: block;
21     width: 50%;
22     height: 100px;
23     color: white; /* Цвет текста */
24     background-color: cornflowerblue; /* Цвет фона */
25 }
```

Результат:



2. display: inline;

Значение **inline** для свойства display задаст элементу следующие настройки:

- элемент будет видимым на странице;
- границы элемента будут определяться его содержимым с учётом переносов на новую строку, **width и height не работают**;
- элементы с display: inline; будут расположены последовательно друг за другом; их поведение похоже на слово в тексте, которое переносится на новую строку, если не умещается в своём блоке.

`display: inline;`

Элемент 1

Элемент 2

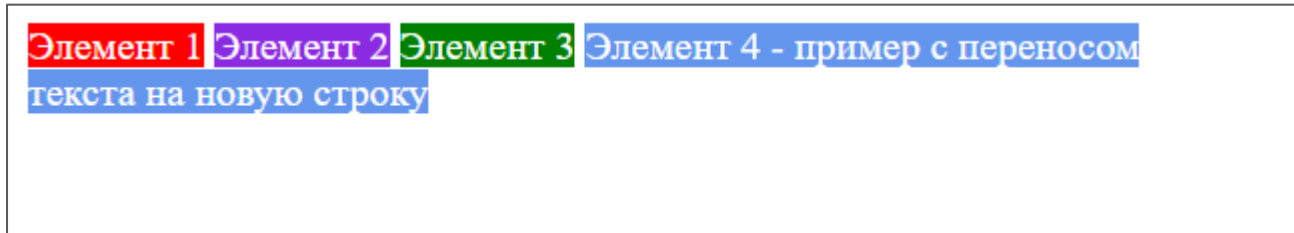
Элемент 3

Код:

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=0.5">
6          <title>Изучаем CSS</title>
7          <link rel="stylesheet" href="css/style.css">
8      </head>
9      <body>
10         <div class="element-1">Элемент 1</div>
11         <div class="element-2">Элемент 2</div>
12         <div class="element-3">Элемент 3</div>
13         <div class="element-4">Элемент 4 - пример с переносом <br /> текста на новую строку</div>
14     </body>
15 </html>
```

```
1  .element-1 {
2      display: inline;
3      color: ■white; /* Цвет текста */
4      background-color: ■red; /* Цвет фона */
5  }
6
7  .element-2 {
8      display: inline;
9      color: ■white; /* Цвет текста */
10     background-color: ■blueviolet; /* Цвет фона */
11 }
12
13 .element-3 {
14     display: inline;
15     color: ■white; /* Цвет текста */
16     background-color: ■green; /* Цвет фона */
17 }
18
19 .element-4 {
20     display: inline;
21     color: ■white; /* Цвет текста */
22     background-color: ■cornflowerblue; /* Цвет фона */
23 }
```

Результат:



3. display: inline-block;

Значение **inline-block** сочетает инлайновое и блочное поведение элемента, а именно:

- элемент будет видимым на странице;
- ширина и высота будут определяться содержимым этого элемента, **width** и **height** работают;
- элементы с display: inline-block; будут расположены последовательно друг за другом; их поведение похоже на слово в тексте, которое переносится на новую строку, если не умещается в своём блоке.

`display: inline-block;`

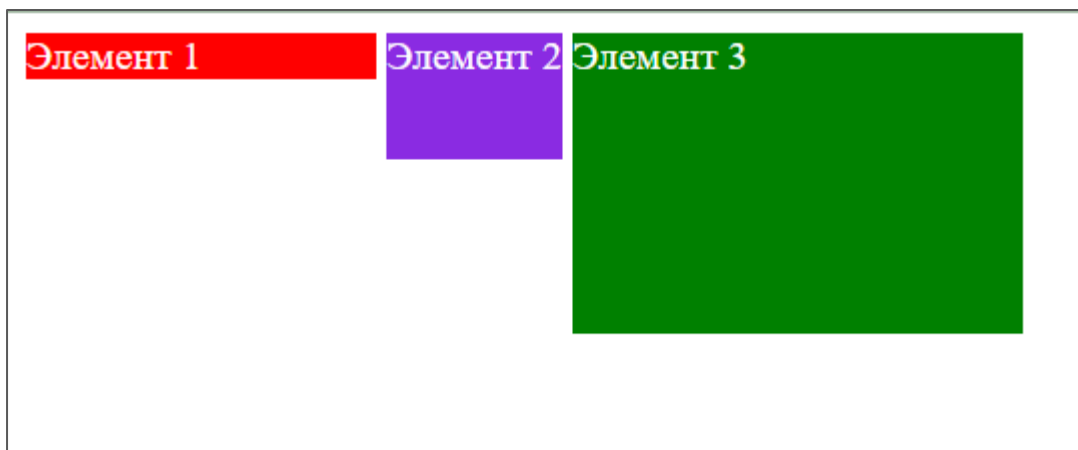


Код:


```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=0.5">
6          <title>Изучаем CSS</title>
7          <link rel="stylesheet" href="css/style.css">
8      </head>
9      <body>
10         <div class="element-1">Элемент 1</div>
11         <div class="element-2">Элемент 2</div>
12         <div class="element-3">Элемент 3</div>
13     </body>
14 </html>
```

```
1  .element-1 {
2      display: inline-block;
3      width: 140px;
4      color: ■ white; /* Цвет текста */
5      background-color: ■ red; /* Цвет фона */
6  }
7
8  .element-2 {
9      display: inline-block;
10     height: 50px;
11     color: ■ white; /* Цвет текста */
12     background-color: ■ blueviolet; /* Цвет фона */
13 }
14
15 .element-3 {
16     display: inline-block;
17     width: 180px;
18     height: 120px;
19     color: ■ white; /* Цвет текста */
20     background-color: ■ green; /* Цвет фона */
21 }
```

Результат:



4. display: none;

Если вы хотите визуально скрыть элемент, более доступной альтернативой является использование комбинации свойств для визуального удаления изображения с экрана, но это сохраняет его для синтаксического анализа с помощью вспомогательных технологий, таких как считыватели экрана.

Значение **none** для свойства display задаст элементу следующие настройки:

- элемент будет **невидимым** на странице;
- элемент не будет влиять на расположение других элементов.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=0.5">
6          <title>Изучаем CSS</title>
7          <link rel="stylesheet" href="css/style.css">
8      </head>
9      <body>
10         <div class="element-1">Элемент 1</div>
11         <div class="element-2">Элемент 2 - скрытый</div>
12         <div class="element-3">Элемент 3</div>
13     </body>
14 </html>
```

```

1  .element-1 {
2      color: ■ white; /* Цвет текста */
3      background-color: ■ red; /* Цвет фона */
4  }
5
6  .element-2 {
7      display: none; /* Скрытый элемент */
8      color: ■ white; /* Цвет текста */
9      background-color: ■ blueviolet; /* Цвет фона */
10 }
11
12 .element-3 {
13     color: ■ white; /* Цвет текста */
14     background-color: ■ green; /* Цвет фона */
15 }

```

Результат:

Элемент 1
Элемент 3

Верстка по сетке

Внимание! Это очень важный раздел, так как созданные образцы в будущем будут основой ваших сайтов

Сетка - это каркас веб-страницы, который нужен для расположения блоков внутри её элементов.

Для чего нужна сетка?

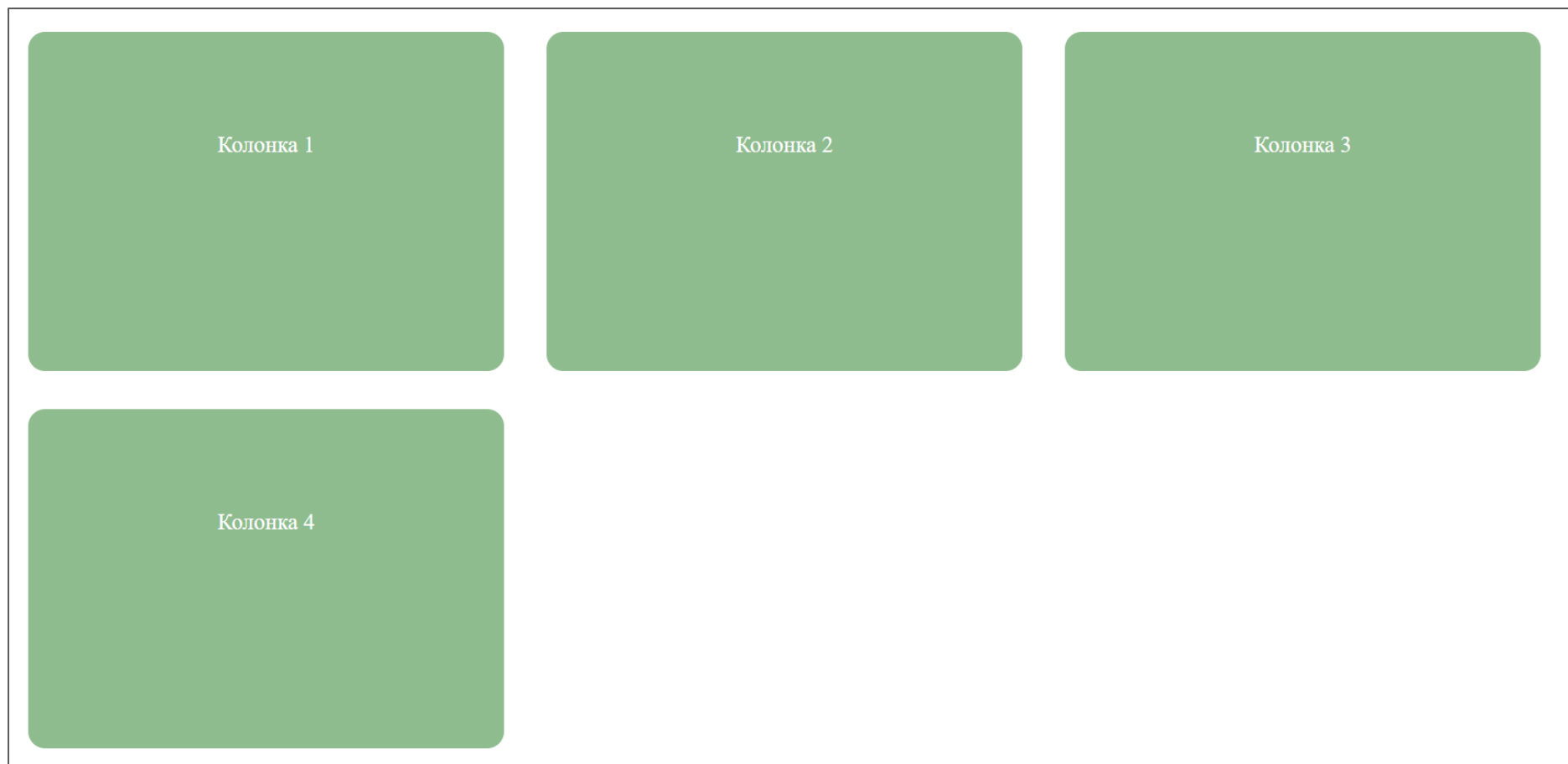
С её помощью дизайнер может организовать элементы макета, привести их к единому формату и определённой логике. Такой подход поможет создать более целостный проект, аккуратный и профессиональный.

1. Создадим четыре элемента `<div>` с классами `col` - это будут колонки сетки. Укажем настройки ширины, высоты и фона. Свойство `display` со значением `inline-block` поможет расположить элементы последовательно слева направо:

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=0.5">
6          <title>Пример создания сетки</title>
7          <link rel="stylesheet" href="css/style.css">
8      <body>
9          <div class="col">Колонка 1</div>
10         <div class="col">Колонка 2</div>
11         <div class="col">Колонка 3</div>
12         <div class="col">Колонка 4</div>
13     </body>
14 </html>|
```

```
1  .col {  
2      display: inline-block; /* Устанавливаем блочный элемент в строку */  
3      width: 25%; /* Ширина блока */  
4      height: 300px; /* Высота блока */  
5      background-color: darkseagreen; /* Цвет фона */  
6      color: white; /* Цвет текста */  
7      font-size: 20px; /* Размер текста */  
8      text-align: center; /* Центрирование текста */  
9      line-height: 200px; /* Центрирование текста по вертикали */  
10     margin: 1%; /* Отступ между блоками */  
11     border-radius: 15px; /* Скругление углов */  
12 }
```

Результат:



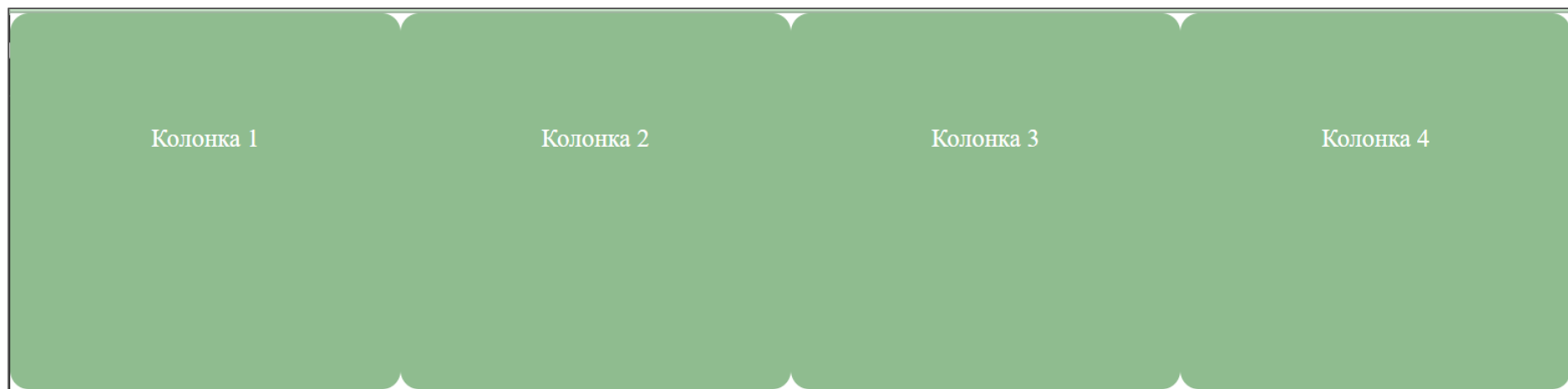
Ширина каждой из четырёх колонок составляет 25%, в сумме они дадут 100% от размера экрана, но одна из колонок по-прежнему смещена и расположена внизу. Обратите внимание: между колонками есть небольшие пробелы. Это связано с тем, что элементы с `display: inline-block` ведут себя почти так же, как символы в тексте. Между символами есть отступы, чтобы они не склеивались **по горизонтали**. Создадим ещё один `<div>` с классом

row. Зададим ему свойство размера шрифта font-size со значением 0, а сами колонки вложим внутрь этого блока. Не забудьте вернуть колонкам размер шрифта font-size: 16px; в настройках класса col, иначе текст в колонке отображаться не будет.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=0.5">
6      <title>Пример создания сетки</title>
7      <link rel="stylesheet" href="css/style.css">
8    <body>
9      <div class="row">
10        <div class="col">Колонка 1</div>
11        <div class="col">Колонка 2</div>
12        <div class="col">Колонка 3</div>
13        <div class="col">Колонка 4</div>
14      </div>
15    </body>
16  </html>
```

```
1  body {
2    margin: 0;
3  }
4
5  .row {
6    font-size: 0;
7  }
8
9  .col {
10   display: inline-block; /* Устанавливаем блочный элемент в строку */
11   width: 25%; /* Ширина блока */
12   height: 300px; /* Высота блока */
13   background-color: darkseagreen; /* Цвет фона */
14   color: white; /* Цвет текста */
15   font-size: 20px; /* Размер текста */
16   text-align: center; /* Центрирование текста */
17   line-height: 200px; /* Центрирование текста по вертикали */
18   margin: 0%; /* Отступ между блоками */
19   border-radius: 15px; /* Скругление углов */
20 }
```

Результат:



Уберите радиус скругления:



Псевдоклассы

Классы удобны для стилизации набора однотипных элементов, которыми и являются колонки сетки. Но что, если для определённых колонок необходимо задать уникальные параметры, с помощью которых нужно, например, изменить цвет фона?

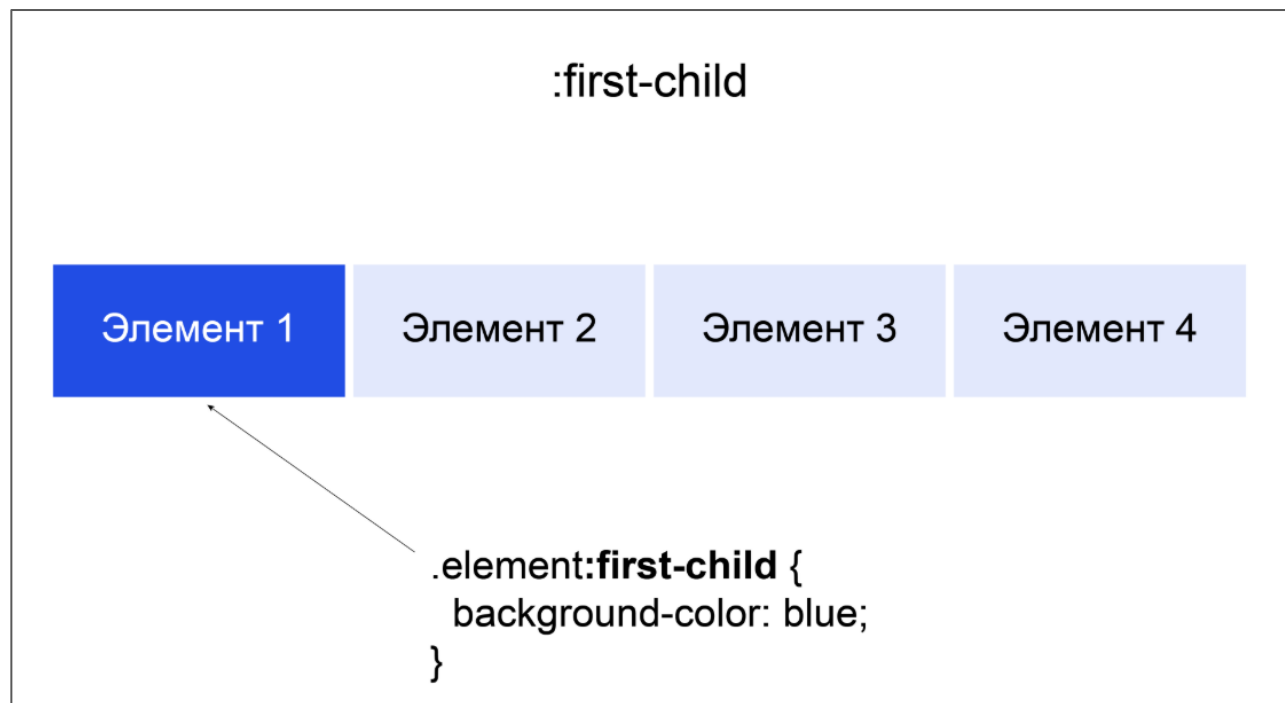
Псевдокласс в CSS — это специальное ключевое слово, добавленное к селектору.

Рассмотрим четыре основных псевдокласса, которые помогут вам стилизовать блоки.

:first-child

:first-child — псевдокласс, который поможет стилизовать **первый элемент родительского блока**. В нашем примере сетки родителем является блок с классом `row`, и с помощью `:first-child`, применённого к `col`, можно изменить цвет фона первой колонки.

Для этого в селекторе указывается `.название-класса:first-child`. Обратите внимание, при создании псевдокласса пробелы не пишутся.



Код:

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=0.5">
6          <title>Пример создания сетки</title>
7          <link rel="stylesheet" href="css/style.css">
8      <body>
9          <div class="row">
10             <div class="col">Колонка 1</div>
11             <div class="col">Колонка 2</div>
12             <div class="col">Колонка 3</div>
13             <div class="col">Колонка 4</div>
14         </div>
15     </body>
16 </html>
```

```
1  body {
2    margin: 0;
3  }
4
5  .row {
6    font-size: 0;
7  }
8
9  .col {
10   display: inline-block;
11   padding: 30px;
12   width: 25%;
13   height: 300px;
14   box-sizing: border-box;
15   font-size: 16px;
16   background-color:  darkseagreen;
17 }
18
19 .col:first-child {
20   background-color:  aquamarine;
21 }
```

Результат:

Колонка 1	Колонка 2	Колонка 3	Колонка 4

:last-child

:last-child работает так же, как и :first-child, только стилизация будет применена к последнему элементу родителя.

Для этого в селекторе указывается .название-класса:**last-child**. Обратите внимание, **при создании псевдокласса пробелы не пишутся.**

:last-child

Элемент 1

Элемент 2

Элемент 3

Элемент 4

```
.element:last-child {  
  background-color: blue;  
}
```



Код (изменения в CSS только):

```
1  body {
2    | margin: 0;
3  }
4
5  .row {
6    | font-size: 0;
7  }
8
9  .col {
10   | display: inline-block;
11   | padding: 30px;
12   | width: 25%;
13   | height: 300px;
14   | box-sizing: border-box;
15   | font-size: 16px;
16   | background-color:  darkseagreen;
17 }
18
19 .col:last-child {
20   | background-color:  aquamarine;
21 }
```

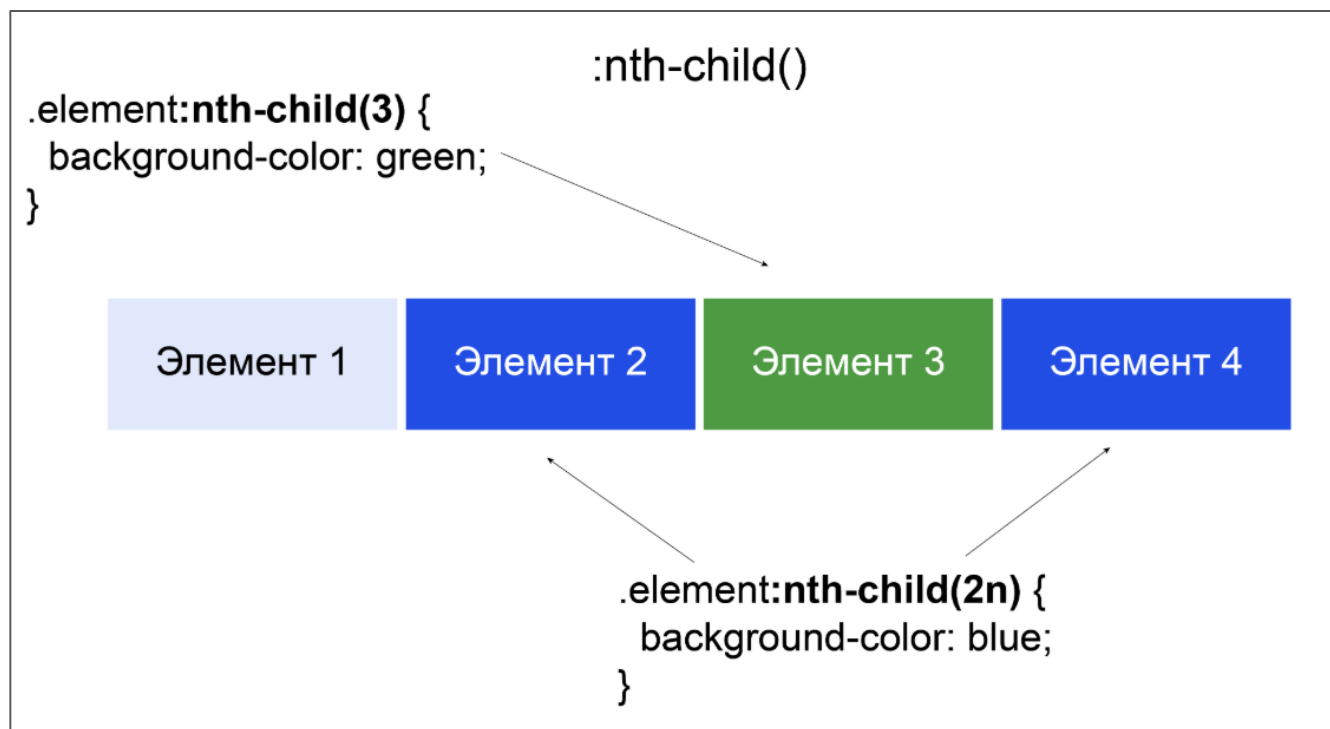
Результат:

Колонка 1	Колонка 2	Колонка 3	Колонка 4

:nth-child()

:nth-child устанавливает стилизацию для элементов с порядковым номером:




- .название-класса:**nth-child(3)** — будет стилизован только третий элемент;
- .название-класса:**nth-child(2n)** — будет стилизован **каждый второй** элемент. Обратите внимание, при создании псевдокласса пробелы не пишутся.

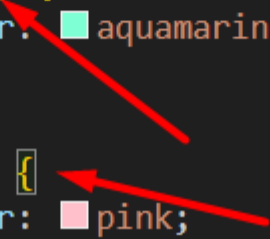


Код (css):

Обратите внимание на `:nth-child(3)` и `:nth-child(2n)`

! Объясните результат.

```
1  body {
2    margin: 0;
3  }
4
5  .row {
6    font-size: 0;
7  }
8
9  .col {
10   display: inline-block;
11   padding: 30px;
12   width: 25%;
13   height: 300px;
14   box-sizing: border-box;
15   font-size: 16px;
16   background-color:  darkseagreen;
17 }
18
19 .col:nth-child(2n) {
20   background-color:  aquamarine;
21 }
22
23 .col:nth-child(3) {
24   background-color:  pink;
25 }
```



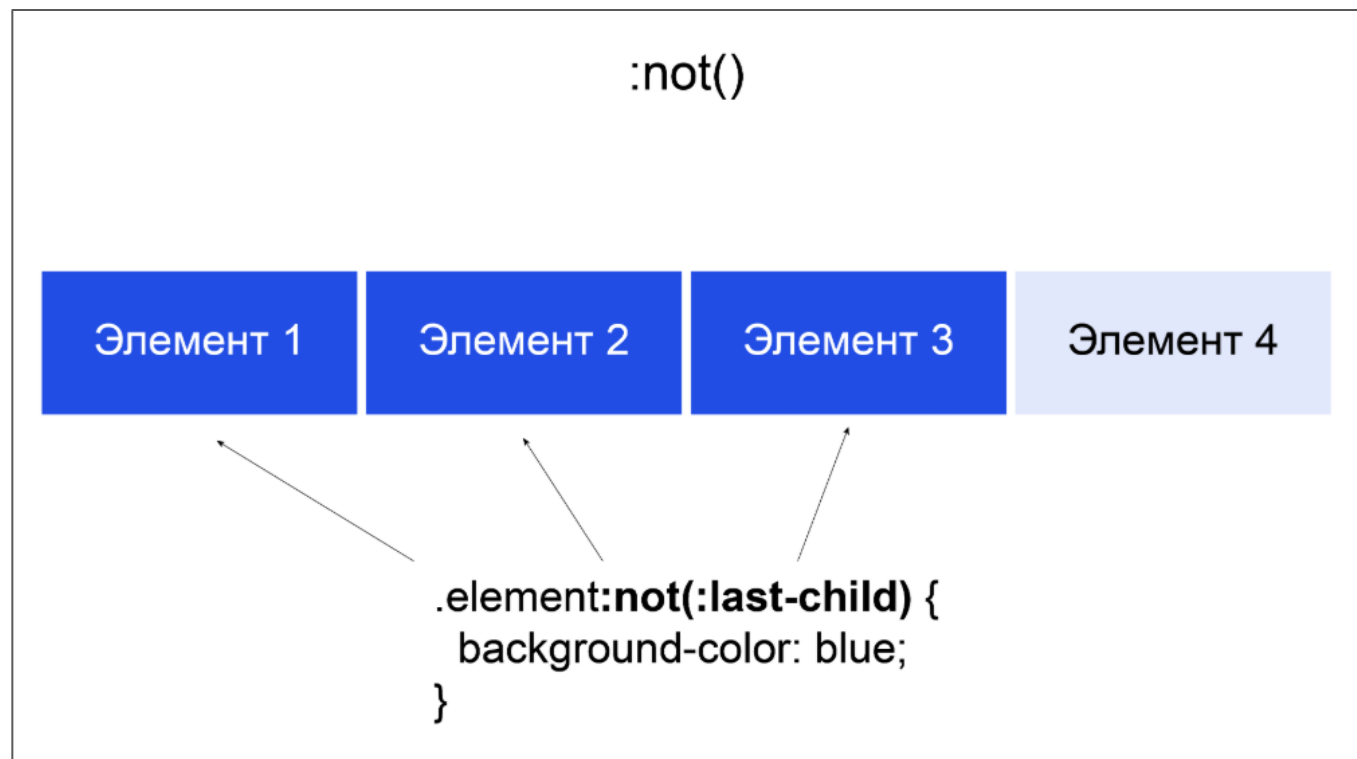
Результат:

Колонка 1	Колонка 2	Колонка 3	Колонка 4
-----------	-----------	-----------	-----------

:not()

:not() используется с другими псевдоселекторами и помогает стилизовать все элементы, кроме тех, которые указаны внутри скобок:

- .название-класса:**not(:first-child)** — будут стилизованы все элементы, кроме первого;
- .название-класса:**not(:last-child)** — будут стилизованы все элементы, кроме последнего;
- .название-класса:**not(:nth-child(2))** — будут стилизованы все элементы, кроме второго. **Обратите внимание, при создании псевдокласса пробелы не пишутся.**



Код:


```
1  body {
2    margin: 0;
3  }
4
5  .row {
6    font-size: 0;
7  }
8
9  .col {
10   display: inline-block;
11   padding: 30px;
12   width: 25%;
13   height: 300px;
14   box-sizing: border-box;
15   font-size: 16px;
16   background-color:  darkseagreen;
17 }
18
19 .col:not(:last-child) {
20   background-color:  pink;
21 }
```

Результат:

Колонка 1	Колонка 2	Колонка 3	Колонка 4

Задания для самостоятельной работы

Реализуйте:

1.

Здесь будет текст.

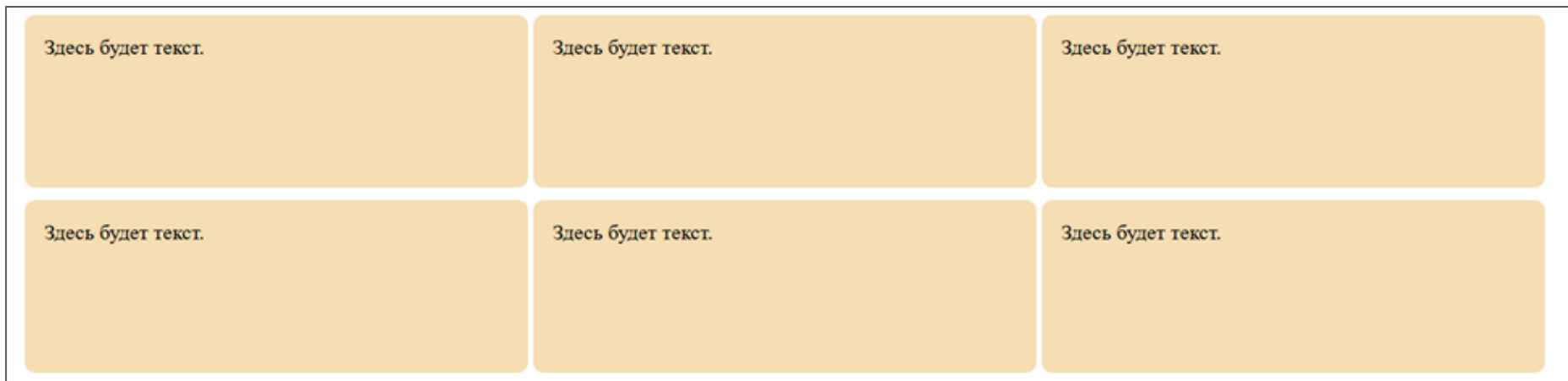
Здесь будет текст.

Здесь будет текст.

Здесь будет текст.

Здесь будет текст.

Здесь будет текст.

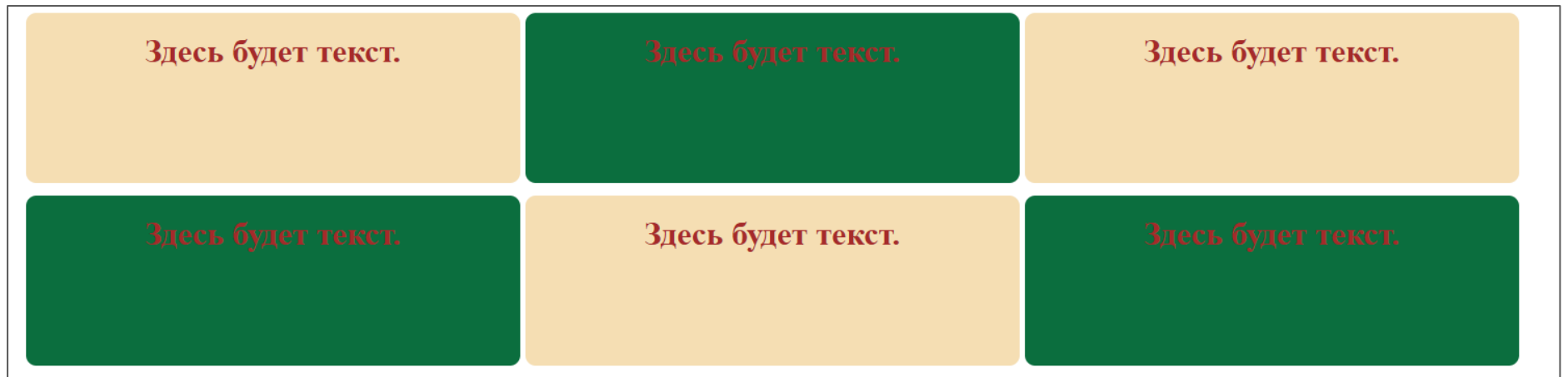


1.

2.



3.



4.



5. Стилизируйте вашу страницу про животных, применив блоки в разных вариантах.