

Модуль 2. Практическая работа №1. Базовые механики построения сеток

Цель работы

Познакомимся с техниками создания крупных сеток страниц и мелких сеток компонентов. Разберём возможности двух технологий создания раскладок: флексбоксов и гридов.

В этой работе мы начнём изучать технологию гридов: разберём, как разными способами создавать грид-раскладки, как управлять расположением грид-элементов в грид-контейнере. Затем мы применим полученные знания на практике и сверстаем с помощью гридов несложный макет.

Результат работы

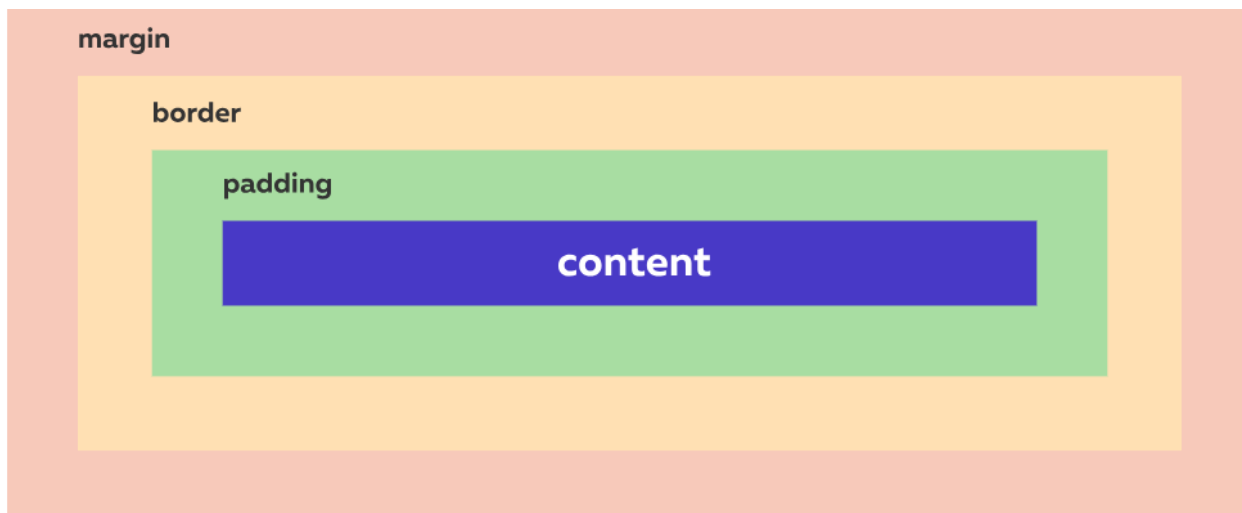
Результат – ссылка на готовую верстку по макету (в codepen или github)
Также приветствуется выполнение дополнительного задания.

Конспект «Сетки»

Бокс

Каждому тегу на странице соответствует прямоугольная область, которая называется **боксом** (от английского *box* — «коробка»).

Бокс состоит из содержимого (content), внутренних отступов (padding), рамки (border) и внешних отступов (margin):



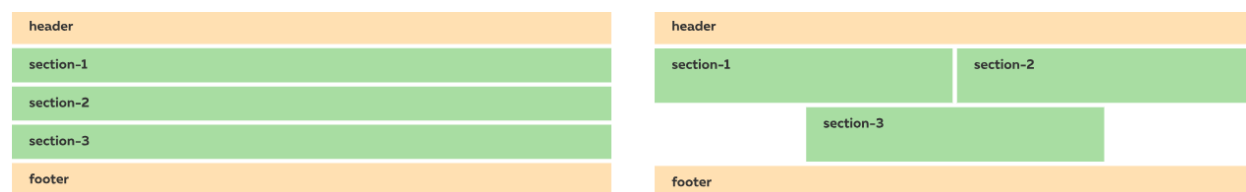
То, как бокс выглядит на странице, во многом зависит от его типа (или от типа его родителя).

Блочные боксы на странице начинаются с новой строки и растягиваются на всю ширину родительского элемента. Блочный тип по умолчанию имеют, например, теги `<p>`, `<div>` и `<h1>`.

Строчные боксы располагаются друг за другом на одной строке, а их ширина зависит от их содержимого. По умолчанию строчными боксами являются, например, теги `<a>`, `` и ``.

Поток, сетки и макет

То, как боксы взаимодействуют друг с другом и в каком порядке располагаются на странице, называется **поток**ом. Поток



Нормальный поток

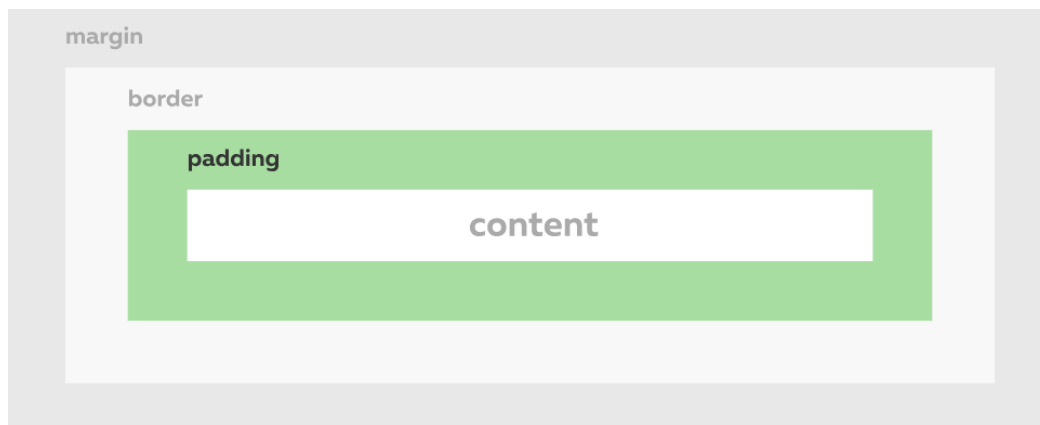
Изменённый поток

Сеткой называют расположение крупных боксов на странице. К таким боксам обычно относят шапку, подвал сайта, основное (`<main>`) и дополнительное (`<aside>`) содержимое, различные секции и разделы. Как правило, количество сеточных элементов на странице не меняется, а их размеры задаются согласно макету.

Макет — это изображение веб-страницы. Его создаёт дизайнер, а веб-разработчик использует его как образец при вёрстке.

Свойство padding

Внутренним отступом называют расстояние между содержимым бокса и рамкой.



Внутренние отступы у элемента создают с помощью свойства `padding`. Если внутренние отступы одинаковы со всех сторон, то достаточно написать так:

```
.element {  
  padding: 15px;  
}
```

Такую запись называют краткой.

Если отступы с разных сторон различаются, то используют полную запись, указывая внутренний отступ отдельно для каждой стороны:

```
.element {  
  padding-top: 5px;  
  padding-right: 10px;  
  padding-bottom: 15px;  
  padding-left: 20px;  
}
```

Свойство `padding-top` создаёт внутренний отступ сверху, `padding-right` — справа, `padding-bottom` — снизу, а `padding-left` — слева.

Свойство `margin`

Внешним отступом называют отступ от внешней границы элемента до границ родительского элемента или до соседних элементов.



Чтобы управлять внешними отступами, используют свойство `margin`. У него, как и у `padding`, есть краткая и полная записи.

```
// Краткая запись  
margin: 20px;
```

```
// Полная запись  
margin-top: 0;  
margin-right: 5px;  
margin-bottom: 10px;  
margin-left: 15px;
```

Свойство `margin-top` создаёт внешний отступ сверху, `margin-right` — справа, `margin-bottom` — снизу, а `margin-left` — слева.

Свойство `display`

За тип бокса в CSS отвечает свойство `display`. У этого свойства больше десятка возможных значений, все они перечислены в [спецификации](#).

```
display: grid;
```

Grid

Бокс с типом `grid` называют грид-контейнером, а дочерние, то есть непосредственно вложенные в него теги — грид-элементами.

Хотя снаружи (для других элементов, например, основного содержимого) грид-контейнер ничем не отличается от блочного бокса, грид-элементы внутри него ведут себя иначе. Например, даже строчные боксы начинают занимать всю доступную им область. Кроме того, в грид-контейнере по-другому ведут себя внешние отступы у элементов.

По умолчанию грид-контейнер одноколонный. Чтобы это изменить, нужно описать шаблон грид-контейнера. Для этого используют свойство `grid-template-columns`:

```
.grid-container {  
  display: grid;  
  grid-template-columns: 100px 150px 80px;  
}
```

Существуют и другие свойства для описания шаблона грид-контейнера. Например, `grid-template-rows` и `grid-template-areas`.

Если элементов в грид-контейнере больше, чем колонок, то следующие элементы автоматически переносятся на новую строку, или ряд, и так же разделяются на колонки.

fr

`fr` (сокращённое от fraction — «доля») — особая единица измерения. Она означает долю доступного пространства в грид-контейнере.

```
.grid-container {  
  display: grid;  
  grid-template-columns: 1fr 2fr;  
}
```

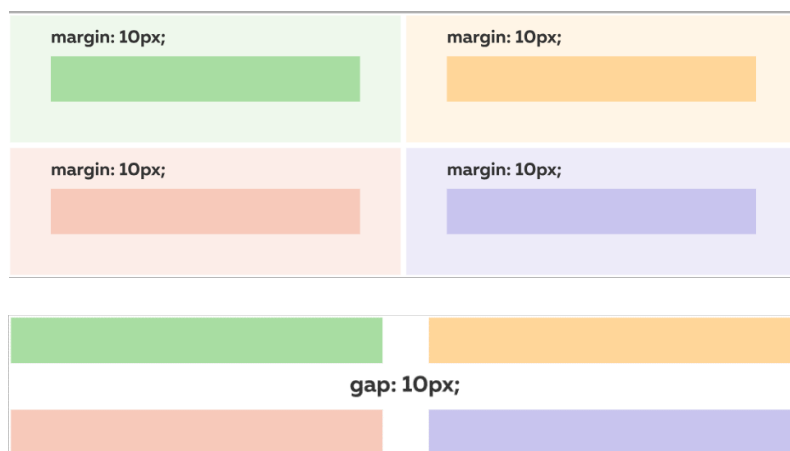
Грид-контейнер в примере будет поделён на 3 равные части. Первая колонка получит одну часть ширины грид-контейнера, а вторая колонка — две части. Как бы ни изменялась ширина контейнера, пропорции колонок всегда будут одинаковыми.

`Fr` можно использовать и вместе с пикселями. Например, вот так можно создать сетку, где правая колонка имеет фиксированную ширину `200px`, а левая занимает всё оставшееся пространство:

```
.grid-container {  
  display: grid;  
  grid-template-columns: 1fr 200px;  
}
```

Свойство `gap`

Свойство `gap` задаёт расстояние между грид-элементами, но не влияет на расстояние между элементами и контейнером. Сравните:



Свойство `gap` добавляется грид-контейнеру, в то время как `margin` — элементам.

С помощью `gap` отступы можно указать отдельно по вертикали и по горизонтали: `column-gap` отвечает за расстояние между колонками, а `row-gap` — за расстояние между рядами.

```
.grid-container {  
  column-gap: 15px;  
  row-gap: 5px;  
}
```

Если же отступы одинаковы, удобно использовать составное свойство `gap`:

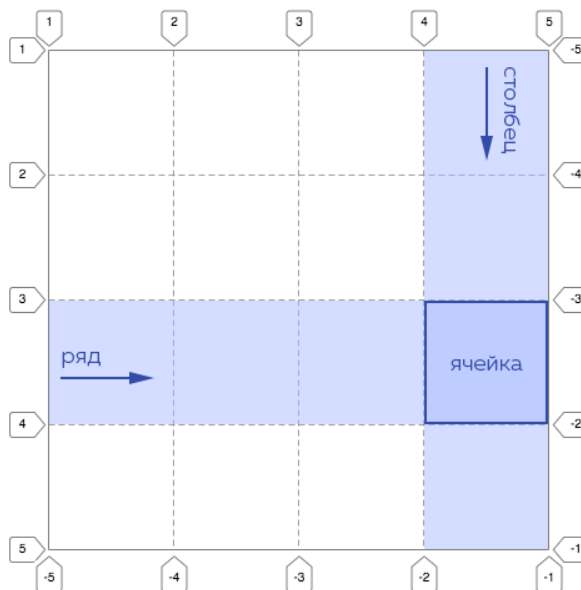
```
.grid-container {  
  gap: 20px;  
}
```

Расположение грид-элементов

Чтобы сделать элемент грид-контейнером, нужно задать ему соответствующее значение свойства `display`:

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

В гриде элементы располагаются по двумерной сетке. То есть грид состоит из рядов и столбцов, располагающихся между линий, которые нумеруются по порядку. Одно «деление» грида называют ячейкой.



Чтобы расположить элемент по сетке внутри грида, нужно задать ему координаты по столбцам и по рядам: с какой линии столбцов и рядов грид-область будет начинаться, на какой линии столбцов и рядов будет заканчиваться. Координаты грид-области на иллюстрации выше в коде описываются так:

```
/*
Область
начинается с 4 линии столбцов,
заканчивается на 5 линии столбцов,
начинается на 3 линии рядов,
заканчивается на 4 линии рядов.
*/

.element {
  grid-column-start: 4;
  grid-column-end: 5;
  grid-row-start: 3;
  grid-row-end: 4;
}
```

Координаты можно отсчитывать не только от начала, но и от конца грида. При этом к индексу линии, от которой ведётся отсчёт, добавляет знак «минус». Координаты той же грид-области можно описать следующим образом:

```
/*
Область
начинается со 2 линии столбцов с конца грида,
заканчивается на 1 линии столбцов с конца грида,
начинается на 3 линии рядов с конца грида,
заканчивается на 2 линии рядов с конца грида.
*/
```

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

Существует также сокращённый синтаксис для этих свойств. Свойство `grid-column` объединяет в себе сразу два свойства: `grid-column-start/grid-column-end`.

Пример:

```
grid-column: 1 / 3;
```

```
/* Это то же самое, что: */
```

```
grid-column-start: 1;
grid-column-end: 3;
```

Аналогично, свойство `grid-row` — это сокращение для задания пары свойств: `grid-row-start/grid-row-end`.

Пример:

```
grid-row: 1 / -2;
```

```
/* Это то же самое, что: */
```

```
grid-row-start: 1;
grid-row-end: -2;
```

Если в свойстве `grid-row` или `grid-column` не задать второй параметр, то значение останется валидным, но применится только первый параметр.

Грид-элементы могут наслаиваться друг на друга, при этом они начинают себя вести как будто абсолютно спозиционированные, при этом на них так же действует свойство `z-index`. Чем больше значение `z-index`, тем выше грид-элемент в «стопке».

Создание раскладки

Чтобы задать гриду определённое количество столбцов и рядов, существуют свойства `grid-template-columns` и `grid-template-rows`.

Свойство `grid-template-columns` перечисляет количество и ширину будущих столбцов грида:

```
/*
Задаём гриду три столбца,
первый шириной 100px,
второй шириной 200px,
третий — 300px.
*/
```

```
.element {  
  grid-template-columns: 100px 200px 300px;  
}
```

Аналогично `grid-template-columns` работает и свойство `grid-template-rows`, только оно сообщает гриду, сколько рядов он будет содержать и какой они будут высоты:

```
/*  
Задаём гриду три ряда,  
первый высотой 100px,  
второй высотой 200px,  
третий – 300px.  
*/
```

```
.element {  
  grid-template-rows: 100px 200px 300px;  
}
```

Также есть возможность задавать нефиксированный размер ячейкам. Для этого существует значение `auto`:

```
/*  
Задаём гриду два столбца,  
первый с нефиксированной шириной,  
второй шириной 100px.  
*/
```

```
.element {  
  grid-template-columns: auto 100px;  
}
```

```
/*  
Задаём гриду три ряда,  
первый высотой 100px,  
второй с нефиксированной высотой,  
третий высотой 200px.  
*/
```

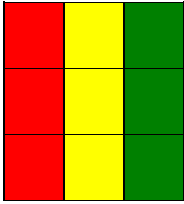
```
.element {  
  grid-template-rows: 100px auto 200px;  
}
```

При заданных свойствах `grid-template-columns` и `grid-template-rows` грид-элементы вписываются в заданную сетку автоматически. При этом часть грид-элементов также может иметь чёткие координаты в гриде. Комбинируя задание явного расположения грид-элементов и их автоматическое распределение, можно строить сложные и одновременно гибкие сетки.

Ещё один механизм создания раскладки грида заключается в использовании свойств `grid-template-areas` и `grid-area`. В значении

свойства `grid-template-areas` визуально «по клеточкам» описывается структура грида.

Пример:



HTML:

```
<div class="grid-container">
  <div class="grid-element-1"></div>
  <div class="grid-element-2"></div>
  <div class="grid-element-3"></div>
</div>
```

CSS:

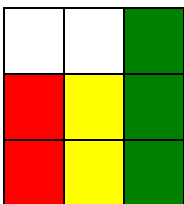
```
.grid-container {
  grid-template-areas:
    "red yellow green"
    "red yellow green"
    "red yellow green";
}
```

```
.grid-element-1 {
  grid-area: red;
}
```

```
.grid-element-2 {
  grid-area: yellow;
}
```

```
.grid-element-3 {
  grid-area: green;
}
```

Свойство `grid-template-areas` позволяет некоторые ячейки грида помечать как пустые. Для этого вместо буквенного именования области используется символ точки.



```
.grid-container {
  grid-template-areas:
```

```

    ". . . green"
    "red yellow green"
    "red yellow green";
}

```

Свойство `gap` позволяет добавлять равномерный интервал между рядами и столбцами. Чтобы добавить интервал только между рядами, используется свойство `row-gap`, а только между столбцами — `column-gap`.

```

.grid-container {
  gap: 10px; /* Между рядами и столбцами интервал 10px */
  column-gap: 20px; /* Между столбцами интервал 20px */
  row-gap: 30px; /* Между рядами интервал 30px */
}

```

Микросетки

Микросетки — сетки мелких элементов веб-страницы.

В отличие от крупных сеток, микросетки меньше зависят от макета и больше — от содержимого. Содержимое страниц со временем может меняться. Если не учитывать этого, вёрстка сломается.

Отступы у ссылок

Часто ссылкам добавляют внутренние отступы, чтобы увеличить область, по которой можно кликнуть (её ещё называют активной областью). Чем проще попасть по ссылке, тем удобнее интерфейс.

По умолчанию ссылки имеют строчный тип бокса. Браузер игнорирует внешние отступы по вертикали у строчных боксов, а их внутренние отступы сверху и снизу не влияют на расположение других элементов. Самый простой способ решить эту проблему — изменить у ссылок тип бокса. Например, сделать их блочными боксами:

```

.element {
  display: block;
}

```

Свойство `align-items`

По умолчанию грид-элементы занимают всё доступное пространство по высоте. Такое поведение можно изменить с помощью свойства `align-items`. Оно задаётся грид-контейнеру и управляет выравниванием грид-элементов по вертикали.

```

.grid-container {
  display: grid;
  align-items: start;
}

```

У `align-items` могут быть следующие значения:

- `stretch` — значение по умолчанию; элементы начинаются у верхней границы и растягиваются на всю высоту.

- `start` — элементы выстраиваются по верхней границе и, если содержимого немного, не тянутся на всю высоту;
- `end` — элементы выстраиваются по нижней границе;
- `center` — элементы располагаются по центру;

Свойство `flex-wrap`

Если элементам не хватает места, они могут вылезти за границы контейнера. Такое поведение называют выпадением элементов. Также говорят, что произошло переполнение.

По умолчанию флекс-контейнер однострочный. Чтобы элементы не выпадали из контейнера, его делают многострочным. Для этого используют свойство `flex-wrap` со значением `wrap`.

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

Список на флексах

Свойство `justify-content` со значением `space-between` заставляет первый и последний элемент прижиматься к границам контейнера. Но если в ряду всего два элемента, то свободного пространства между ними может оказаться слишком много. В этом случае лучше использовать `margin`.

Чтобы убрать лишний отступ у последнего элемента в ряду, используют псевдокласс `:nth-child`. Он позволяет выбрать дочерний элемент по его порядковому номеру:

```
// Выберет второй элемент с классом item
.item:nth-child(2) { ... }
```

```
// Выберет каждый второй элемент с классом item
.item:nth-child(2n) { ... }
```

Если не известно, какой элемент окажется в ряду последним, этот способ не сработает.

`repeat`

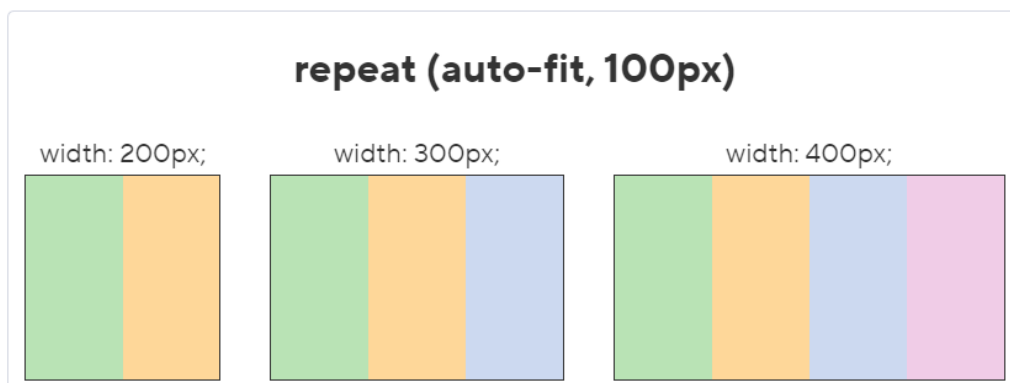
Если все колонки в грид-контейнере должны быть одинаковой ширины, то удобно использовать значение-функцию `repeat`. В скобках после `repeat` указывают количество колонок и их ширину. Значения разделяют запятой:

```
grid-template-columns: repeat(количество колонок, ширина колонки);
```

`auto-fit`

Если количество колонок зависит от ширины контейнера, используют специальное значение `auto-fit`. Его указывают в скобках после `repeat` вместо числа колонок:

```
grid-template-columns: repeat(auto-fit, ширина колонки);
```



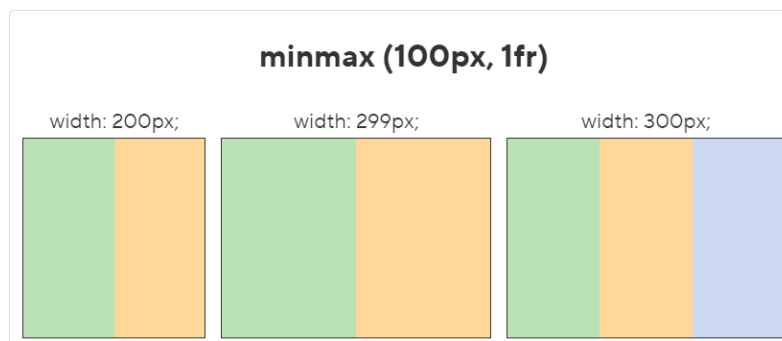
minmax

Чтобы ширина колонок изменялась пропорционально свободному пространству в контейнере, используют значение-функцию `minmax`.

Его указывают в `repeat` вместо фиксированной ширины колонок. В скобках после `minmax` задают минимальный и максимальный размеры колонок, они разделяются запятой:

```
repeat(auto-fit, minmax(минимальный размер, максимальный размер));
```

В `minmax` в качестве максимального значения часто используют единицу измерения `fr`. Она позволяет колонкам увеличивать ширину до тех пор, пока свободного пространства в контейнере не хватит на ещё одну колонку.



Свойства `grid-column` и `grid-row`

Чтобы растянуть элемент на несколько колонок используют свойство `grid-column`. Число после ключевого слова `span` указывает число колонок, которые элемент должен занять:

```
.element {
  grid-column: span 2;
}
```

Растянуть элемент на несколько рядов можно с помощью свойства `grid-row`. Ключевое слово `span` в нём означает количество рядов, которые элемент должен занять:

```
.long-element {
  grid-row: span 2;
}
```

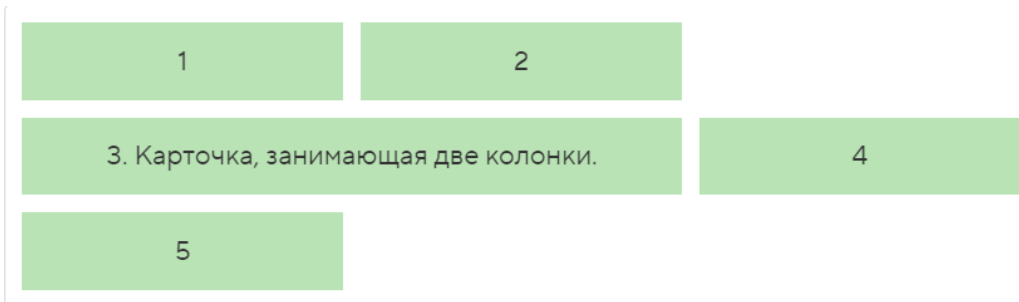
Свойства `grid-column` и `grid-row` можно использовать одновременно.

Свойство `grid-auto-flow`

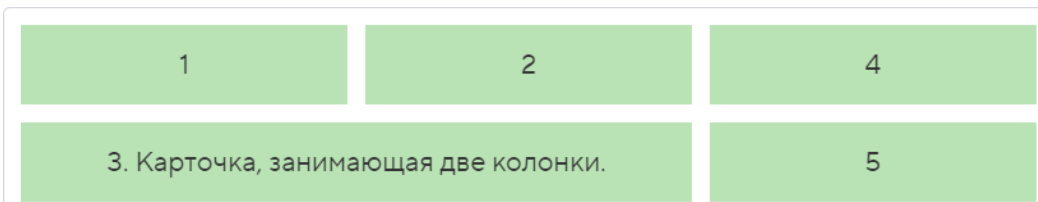
Свойство `grid-auto-flow` управляет автозаполнением грид-контейнера.

```
.grid-container {  
  display: grid;  
  grid-auto-flow: row;  
}
```

Значение по умолчанию `row` говорит располагать элементы в том порядке, в котором они идут в разметке, и при необходимости создавать новые ряды:



Но если указать значение `dense`, то контейнер будет заполняться так, чтобы не было пропусков:



Значение `dense` заставляет грид-контейнер заполнять пустые ячейки первым подходящим по размеру грид-элементом. При этом визуальный порядок на странице может отличаться от порядка элементов в разметке. Если порядок элементов важен, лучше это значение не использовать.

Верстка макета (легкий уровень)

Разбирается макет страницы со статьёй на сайте о зрелищных видах искусства «Performance». Эта статья скорее всего не является главной страницей сайта.

[Макет](#) для кейса предоставлен в формате Figma:

<https://www.figma.com/file/Hr1nFDc1l1FtWb0vd4z4tD/performance?type=design&node-id=0%3A1&mode=design&t=qvC8WGFxt8LqYQnf-1>

Макет состоит из шапки, собственно статьи и подвала. Шапка и подвал не идентичны. У статьи есть своя заголовочная часть, в которую попадают заголовок и метаданные, заголовочная часть на четверть уже основного

контейнера статьи. Метаданные неоднородные, и скорее всего не являются списком определений. Внутри статьи есть фотографии с подписями, их нужно правильным образом компоновать.

В демонстрации мы подробно разберём как построить сетку для страницы с помощью *CSS Layout Grid*. Построим сетки для каркаса сайта и отдельных элементов. Выясним, в каких случаях лучше не использовать сетки на гридах, а ограничиться флексами.

1. Приступим к созданию сетки сайта новостей из мира искусства «Performance». На первом шаге добавим тег meta, чтобы сайт «правильно» отображался и на мобильных устройствах. Укажем заголовок для сайта. Подключим необходимые шрифты. Сразу подключим **стилевой файл**.

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="UTF-8">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link
href="https://fonts.googleapis.com/css2?family=Lora:wght@400;500&display=swap"
rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <title>Главная страница новостного сайта Performance</title>
  </head>
  <body>

  </body>
</html>
```

Не станем слишком подробно останавливаться на семантических вопросах, их мы подробно разбирали в пр. работе 9. «Создание семантической разметки по макету», но некоторые моменты опишем в комментариях к разметке. Сразу добавим обёртки, которые точно понадобятся, а также напомним классы.

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="UTF-8">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link
href="https://fonts.googleapis.com/css2?family=Lora:wght@400;500&display=swap"
rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <title>Главная страница новостного сайта Performance</title>
  </head>
  <body>
    <header class="header">
      <nav class="header-nav container">
        <a class="logo" href="index.html">
          
        </a>
        <ul class="menu">
          <li class="menu-item">
            <a class="menu-item-link" href="#">Премьеры</a>
          </li>
```

```

<li class="menu-item">
  <a class="menu-item-link" href="#">Интервью</a>
</li>
<li class="menu-item">
  <a class="menu-item-link" href="#">Новости</a>
</li>
<li class="menu-item">
  <a class="menu-item-link" href="#">Дискуссии</a>
</li>
<li class="menu-item">
  <a class="menu-item-link" href="#">Рекомендации</a>
</li>
</ul>
</nav>
</header>
<main class="main container">
  <article class="article">
    <header class="article-header">
      <h1 class="article-title">Премьера вызывающе красивого спектакля от режиссера
Жана Портье</h1>
      <div class="article-meta">
        <time class="article-meta-date" datetime="2020-02-20">20.02.2020</time>
        <p class="article-meta-author">Автор статьи: Кристина Петрова</p>
        <p class="article-meta-photograph">Фотограф: Пьер Буше</p>
      </div>
      <!-- Элементы неоднородные, данных о том, как будут выглядеть мета-данные в других
статьях, пока что нет. Обёртка понадобится в любом случае. -->
    </header>
    
    <p class="article-text">Лидер молодой французской режиссуры, создатель и
руководитель независимого театра Emigre выпустил премьеру в театре Эберто 12 февраля.
Жан Портье предлагает зрителю физически ощутить ход времени и поиграть с культурными
кодами: окунуться в таинственную атмосферу начала XIX века и разгадать мистическую,
но в то же время романтическую загадку. «В прошлом году в Монпелье» – вызывающе
красивый, изящный спектакль. Постановка сильно отличается от других, минималистичных,
работ Портье. Здесь же можно увидеть совсем иную атмосферу, изысканную и дорогую.
Двухчасовой спектакль в основе которого старинная мистическая история, где правду
невозможно отличить от вымысла.</p>
    <figure class="article-figure article-figure-horizontal article-figure-wide">
      
      <figcaption class="article-figcaption">
        <p>На фото: режиссер Жан Портье на репетиции спектакля «В прошлом году в
Монпелье»</p>
        <p>Фотограф: Пьер Буше</p>
      </figcaption>
      <!-- То, что в этой статье подписи сделаны однообразно, не говорит о том, что это
список определений. Пока у нас нет данных о том, как это будет сделано в других
статьях, не будем усложнять. -->
    </figure>
    <p class="article-text">В дорогом отеле мужчина уверяет женщину, что они уже
встречались год назад в Монпелье. Женщина считает, что этого не было, либо она просто
не хочет этого вспоминать. Еще один герой – муж или спутник женщины, седой, взрослый
пятидесятилетний мужчина, богатый, молчаливый и, вероятно, делающий возможным ее
красивую праздную жизнь. Отношения троицы не определены, причинно-следственные связи
нарушены, запутанность сюжета вначале кажется абсурдной. Все подчеркивает, как
странно устроен этот мир, давая возможность бесконечного множества трактовок, включая
интерпретацию с помощью мифа об Орфее и Эвридике, представление об отеле как о
чистилище.</p>
    <figure class="article-figure article-figure-wide">

```

```

    
    <figcaption class="article-figcaption">
        <p>На фото: главные актеры спектакля Обен Шарль и Мари Брюней</p>
        <p>Фотограф: Пьер Буше</p>
    </figcaption>
</figure>
<figure class="article-figure article-figure-wide">
    
    <figcaption class="article-figcaption">
        <p>На фото: главные актеры спектакля Обен Шарль и Мари Брюней</p>
        <p>Фотограф: Пьер Буше</p>
    </figcaption>
</figure>
<figure class="article-figure">
    
    <figcaption class="article-figcaption">
        <p>На фото: главная актриса Мари Брюней</p>
        <p>Фотограф: Пьер Буше</p>
    </figcaption>
</figure>
<p class="article-text">Жан Портье предлагает два варианта взаимодействия со
спектаклем: «Или зритель постарается реконструировать некую схему, самую линейную из
всех ему доступных, выстроить связи, пытаться понять логически построение сюжета и
поведение героев – и тогда он найдет эту постановку трудной, если не вовсе
недоступной пониманию, абсурдной. Или же, напротив, позволит увлечь себя необычными
образами, возникшими перед ним силою голосов актеров, различным шумами, музыкой,
ритмом или страстностью героев. Такому зрителю спектакль покажется самым легким из
всех увиденных, адресованным лишь его чувствам – способности видеть, слышать,
ощущать, чувствовать и волноваться».</p>
<figure class="article-figure">
    
    <figcaption class="article-figcaption">
        <p>На фото: главная актриса Мари Брюней</p>
        <p>Фотограф: Пьер Буше</p>
    </figcaption>
</figure>
<figure class="article-figure">
    
    <figcaption class="article-figcaption">
        <p>На фото: Мари Брюней и Обен Шарль</p>
        <p>Фотограф: Пьер Буше</p>
    </figcaption>
</figure>
</article>
</main>
<footer class="footer">
    <div class="footer-content container">
<!-- Подвалу нужна контентная обёртка, мы очень подробно обсуждали это в методике, на
первом шаге. У шапки контейнером выступал тег <nav>, здесь придётся сделать <div>.
При этом мы наверняка знаем, что футеру понадобится сетка, стили которой нельзя
прикрепить к классу container, так что второй класс у тега <div> тоже появится сразу.
-->
        <a class="logo" href="index.html">
            
        </a>
        <p class="footer-copyright">© 2021 Все права защищены</p>
    </div>
</footer>

```



```
</div>
</footer>
</body>
</html>
```

Зададим всему сайту минимальную ширину и высоту не меньше высоты экрана, сбросим лишний внешний отступ. Добавим параметры шрифта. Определим доступное скрытие. Жёстко ограничим и центрируем контентные крупные блоки, для этого у нас есть класс `container`, который мы придали и шапке, и основному содержимому, и футеру. Подключим подсветку, которая будет выделять те блоки, с которыми мы будем иметь дело на каждом шаге.

style.css

```
body {
  min-width: 1440px;
  min-height: 100vh;
  margin: 0;
  font-family: "Lora", "Palatino Linotype", serif;
  font-size: 18px;
  line-height: 1.5;
  color: #1f1e1c;
}

.visually-hidden {
  position: absolute;
  width: 1px;
  height: 1px;
  margin: -1px;
  padding: 0;
  overflow: hidden;
  border: 0 none;
  clip-path: inset(100%);
}

.container {
  width: 1240px;
  margin: 0 auto;
}
```

Несмотря на то, что крупные блоки сайта и сами встали по порядку, будет лучше определить их место, организовать их в сетку. Это сделает систему и более управляемой, и устойчивой. Для этого сделаем сайт гридом, в котором шапка и подвал займут достаточно места для себя, а остальное пространство займёт статья. Теперь, даже если статья будет совсем короткой, шапка и подвал всё равно не изменят высоту и останутся прижатыми к краям экрана.

```
body {
  display: grid;
  grid-template-rows: auto 1fr auto;

  min-width: 1440px;
  min-height: 100vh;
  margin: 0;
  font-family: "Lora", "Palatino Linotype", serif;
  font-size: 18px;
  line-height: 1.5;
  color: #1f1e1c;
}
```

У сайта достаточно скромная навигация: только логотип и меню. Меню будет флексбоксом, потому что нужно учесть его переполнение и возможное увеличение количества пунктов, и при этом сделать размер пунктов зависящим от контента, а грид такое не умеет. Сразу зададим параметры шрифта для меню, опишем его флекс, сбросим лишние отступы. Подробности про флексы — в следующей практической работе — «Построение сеток на флексах по макету». Выстроить логотип и меню в сетку, выровнять их и «прилепить» меню вправо можно гридом.

Как это сделали бы вы?

```
<header class="header">
  <nav class="header-nav container">
    <a class="logo" href="index.html">
      
    </a>
    <ul class="menu">
      <li class="menu-item">
        <a class="menu-item-link" href="#">Премьеры</a>
      </li>
      <li class="menu-item">
        <a class="menu-item-link" href="#">Интервью</a>
      </li>
      <li class="menu-item">
        <a class="menu-item-link" href="#">Новости</a>
      </li>
      <li class="menu-item">
        <a class="menu-item-link" href="#">Дискуссии</a>
      </li>
      <li class="menu-item">
        <a class="menu-item-link" href="#">Рекомендации</a>
      </li>
    </ul>
  </nav>
</header>
```

Сразу добавим пунктам меню отступы и уберём стили ссылок по умолчанию. Меню сейчас выглядит почти как нужно, осталось поставить его на правильное место.

```
.menu {
  display: flex;
  flex-wrap: wrap;

  margin: 0;
  padding: 0;

  list-style: none;
  color: #000000;
  font-size: 16px;
  font-weight: 500;
  line-height: 1.5;
}

.menu-item {
  margin-right: 30px;
}
```

```
.menu-item:last-child {
  margin-right: 0;
}

.menu-item-link {
  color: inherit;
  text-decoration: none;
}
```

Мы сделали двухколоночный грид, в котором у логотипа чёткий размер (логотипы не меняются, поэтому делать их гибко — нецелесообразно), а всё остальное пространство займёт меню.

```
...
/* шапка */
.header-nav {
  display: grid;
  grid-template-columns: 145px 1fr;
  grid-column-gap: 50px;
}

.menu {
  display: flex;
  flex-wrap: wrap;

  margin: 0;
  padding: 0;

  list-style: none;
  color: #000000;
  font-size: 16px;
  font-weight: 500;
  line-height: 1.5;
}

.....
```

Чтобы шапка как контейнер воспринимала размеры правильно, поменяли ей блочную модель (подробнее в материалах по блочной модели). Высоту шапки мы указали так, чтобы она не сжималась меньше 80 пикселей, но в случае, если меню станет двухстрочным, шапка бы не переполнилась.

```
/* шапка */
.header-nav {
  display: grid;
  grid-template-columns: 145px 1fr;
  grid-column-gap: 50px;
  box-sizing: border-box;
  min-height: 80px;
}
```

Меню выравнивается по строчной оси в конец грида. По блочной оси все элементы грида выравниваются по центру. Внутренние отступы сверху и снизу мы добавили из предусмотрительности. Это нужно для того, чтобы, если меню выстроится в две строки, текст не прилип бы к краям шапки.

```
/* шапка */
.header-nav {
  display: grid;
  grid-template-columns: 145px 1fr;
  grid-column-gap: 50px;
  align-items: center;
```

```
padding-top: 25px;
padding-bottom: 25px;
box-sizing: border-box;
min-height: 80px;
}

.menu {
display: flex;
flex-wrap: wrap;
justify-self: end;

margin: 0;
padding: 0;

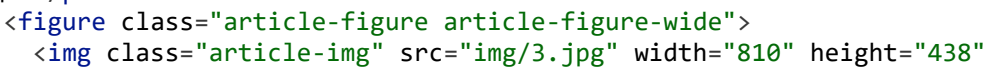
list-style: none;
color: #000000;
font-size: 16px;
font-weight: 500;
line-height: 1.5;
}
```

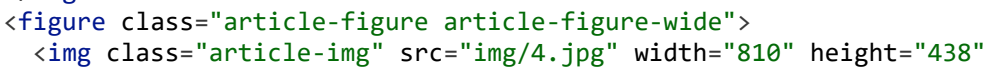
Нас ждёт основное содержимое сайта. Оно, конечно, также будет гридом. Но как мы расположим элементы сайта? Проанализируем макет. Попробуйте нарисовать схему грида на бумаге.

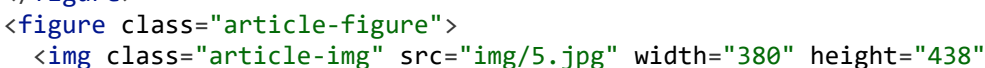
Что у вас получилось?

```
<main class="main container">
  <article class="article">
    <header class="article-header">
      <h1 class="article-title">Премьера вызывающе красивого спектакля от
режиссера Жана Портье</h1>
      <div class="article-meta">
        <time class="article-meta-date" datetime="2020-02-20">20.02.2020</time>
        <p class="article-meta-author">Автор статьи: Кристина Петрова</p>
        <p class="article-meta-photograph">Фотограф: Пьер Буше</p>
      </div>
    </header>
    
    <p class="article-text">Лидер молодой французской режиссуры, создатель и
руководитель независимого театра Emigre выпустил премьеру в театре Эберто 12 февраля.
Жан Портье предлагает зрителю физически ощутить ход времени и поиграть с культурными
кодами: окунуться в таинственную атмосферу начала XIX века и разгадать мистическую,
но в то же время романтическую загадку. «В прошлом году в Монпелье» – вызывающе
красивый, изящный спектакль. Постановка сильно отличается от других, минималистичных,
работ Портье. Здесь же можно увидеть совсем иную атмосферу, изысканную и дорогую.
Двухчасовой спектакль в основе которого старинная мистическая история, где правду
невозможно отличить от вымысла.</p>
    <figure class="article-figure article-figure-horizontal article-figure-wide">
      
      <figcaption class="article-figcaption">
        <p>На фото: режиссер Жан Портье на репетиции спектакля «В прошлом году в
Монпелье»</p>
        <p>Фотограф: Пьер Буше</p>
      </figcaption>
    </figure>
    <p class="article-text">В дорогом отеле мужчина уверяет женщину, что они уже
встречались год назад в Монпелье. Женщина считает, что этого не было, либо она просто
не хочет этого вспоминать. Еще один герой – муж или спутник женщины, седой, взрослый
пятидесятилетний мужчина, богатый, молчаливый и, вероятно, делающий возможным ее
```

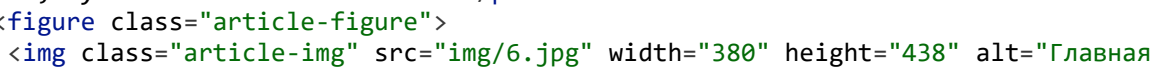
красивую праздную жизнь. Отношения троицы не определены, причинно-следственные связи нарушены, запутанность сюжета вначале кажется абсурдной. Все подчеркивает, как странно устроен этот мир, давая возможность бесконечного множества трактовок, включая интерпретацию с помощью мифа об Орфее и Эвридике, представление об отеле как о чистилище.



На фото: главные актёры спектакля Обен Шарль и Мари Брюней
Фотограф: Пьер Буше


На фото: главные актёры спектакля Обен Шарль и Мари Брюней
Фотограф: Пьер Буше


На фото: главная актриса Мари Брюней
Фотограф: Пьер Буше

Жан Портье предлагает два варианта взаимодействия со спектаклем: «Или зритель постарается реконструировать некую схему, самую линейную из всех ему доступных, выстроить связи, пытаться понять логически построение сюжета и поведение героев – и тогда он найдет эту постановку трудной, если не вовсе недоступной пониманию, абсурдной. Или же, напротив, позволит увлечь себя необычными образами, возникшими перед ним силою голосов актёров, различным шумом, музыкой, ритмом или страстностью героев. Такому зрителю спектакль покажется самым легким из всех увиденных, адресованным лишь его чувствам – способности видеть, слышать, ощущать, чувствовать и волноваться».


На фото: главная актриса Мари Брюней
Фотограф: Пьер Буше


На фото: Мари Брюней и Обен Шарль
Фотограф: Пьер Буше

У нас получился трёхколоночный грид с интервалами по 50 пикселей между колонками и по 70 — между рядами. Все грид-элементы заняли по одной ячейке.

```
/* основное содержимое */
.article {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 70px 50px;
}
```

Зададим расположение тем элементам, которые займут больше одной колонки.

```
/* основное содержимое */
.article {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 70px 50px;
}
```

```
.article-header {
  grid-column: 1 / -1;
}
```

```
.article-img-wide {
  grid-column: span 3;
}
```

```
.article-figure-wide {
  grid-column: span 2;
}
```

Сразу укажем параметры шрифтов у элементов внутри основного содержимого, которые отличаются от указанных в body. Это позволит не отвлекаться на них в дальнейшем, а ещё — лучше видеть, как работает сетка, ведь размер шрифта теперь не изменится.

```
.article-header {
  grid-column: 1 / -1;
}
.article-title {
  font-size: 46px;
  font-weight: normal;
  line-height: 1.5;
  text-align: center;
}
```

```
.article-meta {
  font-size: 18px;
  line-height: 1.5;
}
```

```
.article-figcaption {
  font-size: 16px;
  line-height: 1.5;
}
```

```
.article-img-wide {
  grid-column: span 3;
}
```

Сетка статьи сложилась, и теперь мы можем заниматься частными блоками и элементами. Начнём с шапки статьи. Мы могли бы указать её конкретную ширину, но лучше указать максимальную ширину блока в процентах от ширины родительского контейнера. Следующим шагом мы должны будем задать сетку метаданным.

```
.article-header {
  grid-column: 1 / -1;
  max-width: 75%;
}

.article-title {
  font-size: 46px;
  font-weight: normal;
  line-height: 1.5;
  text-align: center;
}

.article-meta {
  display: grid;
  grid-template-columns: auto auto auto;
  grid-column-gap: 50px;

  font-size: 18px;
  line-height: 1.5;
}
```

Это грид на три колонки. И сделаем между колонками интервал, его размер мы увидели в макете.

Сейчас нужно всё выровнять, и в основном это можно сделать средствами грида. Выравниваем внутри грид-контейнера сам хедер статьи, сбросим отступы по умолчанию у мета-элементов и выравниваем их к центру.

```
.article-header {
  grid-column: 1 / -1;
  max-width: 75%;
  justify-self: center;
}

.article-title {
  font-size: 46px;
  font-weight: normal;
  line-height: 1.5;
  text-align: center;
}

.article-meta {
  display: grid;
  grid-template-columns: auto auto auto;
  grid-column-gap: 50px;
  justify-content: center;

  font-size: 18px;
  line-height: 1.5;
}

.article-meta-author {
  margin: 0;
}
```

```
.article-meta-photograph {
  margin: 0;
}
```

Пора заняться собственно статьёй. Мы подробно разберем работу с текстами далее, так что сейчас остановимся на сеточных сложностях. И начнём с того, что, фотография ведёт себя неправильно, и этот баг нужно исправить. Запретим элементу искажение и попробуем выровнять.

```
<main>
  <article>
    <header>...</header>
    
    <p class="article-text">Лидер молодой французской режиссуры, создатель и
руководитель независимого театра Emigre выпустил премьеру в театре Эберто 12 февраля.
Жан Портье предлагает зрителю физически ощутить ход времени и поиграть с культурными
кодами: окунуться в таинственную атмосферу начала XIX века и разгадать мистическую,
но в то же время романтическую загадку. «В прошлом году в Монпелье» – вызывающе
красивый, изящный спектакль. Постановка сильно отличается от других, минималистичных,
работ Портье. Здесь же можно увидеть совсем иную атмосферу, изысканную и дорогую.
Двухчасовой спектакль в основе которого старинная мистическая история, где правду
невозможно отличить от вымысла.</p>
    <figure class="article-figure article-figure-horizontal article-figure-wide">
      
      <figcaption class="article-figcaption">
        <p>На фото: режиссер Жан Портье на репетиции спектакля «В прошлом году в
Монпелье»</p>
        <p>Фотограф: Пьер Буше</p>
      </figcaption>
    </figure>
    <p class="article-text">В дорогом отеле мужчина уверяет женщину, что они уже
встречались год назад в Монпелье. Женщина считает, что этого не было, либо она просто
не хочет этого вспоминать. Еще один герой – муж или спутник женщины, седой, взрослый
пятидесятилетний мужчина, богатый, молчаливый и, вероятно, делающий возможным ее
красивую праздную жизнь. Отношения троицы не определены, причинно-следственные связи
нарушены, запутанность сюжета вначале кажется абсурдной. Все подчеркивает, как
странно устроен этот мир, давая возможность бесконечного множества трактовок, включая
интерпретацию с помощью мифа об Орфее и Эвридике, представление об отеле как о
чистилище.</p>
    <figure class="article-figure article-figure-wide">
      
      <figcaption class="article-figcaption">
        <p>На фото: главные актеры спектакля Обен Шарль и Мари Брюней</p>
        <p>Фотограф: Пьер Буше</p>
      </figcaption>
    </figure>
    <figure class="article-figure article-figure-wide">
      
      <figcaption class="article-figcaption">
        <p>На фото: главные актеры спектакля Обен Шарль и Мари Брюней</p>
        <p>Фотограф: Пьер Буше</p>
      </figcaption>
    </figure>
  </figure class="article-figure">
```



```

    
    <figcaption class="article-figcaption">
        <p>На фото: главная актриса Мари Брюней</p>
        <p>Фотограф: Пьер Буше</p>
    </figcaption>
</figure>
<p class="article-text">Жан Портье предлагает два варианта взаимодействия со
спектаклем: «Или зритель постарается реконструировать некую схему, самую линейную из
всех ему доступных, выстроить связи, пытаться понять логически построение сюжета и
поведение героев – и тогда он найдет эту постановку трудной, если не вовсе
недоступной пониманию, абсурдной. Или же, напротив, позволит увлечь себя необычными
образами, возникшими перед ним силою голосов актеров, различным шумами, музыкой,
ритмом или страстностью героев. Такому зрителю спектакль покажется самым легким из
всех увиденных, адресованным лишь его чувствам – способности видеть, слышать,
ощущать, чувствовать и волноваться».</p>
<figure class="article-figure">
    
    <figcaption class="article-figcaption">
        <p>На фото: главная актриса Мари Брюней</p>
        <p>Фотограф: Пьер Буше</p>
    </figcaption>
</figure>
<figure class="article-figure">
    
    <figcaption class="article-figcaption">
        <p>На фото: Мари Брюней и Обен Шарль</p>
        <p>Фотограф: Пьер Буше</p>
    </figcaption>
</figure>
</article>

```

```

...
.article-meta-photograph {
    margin: 0;
}

```

```

.article-img {
    vertical-align: middle;
    max-width: 100%;
    height: auto;
}

```

```

.article-figcaption {
    font-size: 16px;
    line-height: 1.5;
}

```

У одного из <figure> есть особенность: он занимает две колонки, и при этом картинка и подпись выстроены в две колонки. Выстроим их в сетку из двух колонок с интервалом, который совпадает с интервалом в гриде статьи.

```

...
.article-figcaption {
    font-size: 16px;
    line-height: 1.5;
}

```

```

.article-figure-horizontal {
    display: grid;
    grid-template-columns: 1fr 1fr;
}

```

```
gap: 50px;
}
```

```
.article-img-wide {
  grid-column: span 3;
}
```

Сейчас должно быть особенно заметно, что у большинства элементов, которые встроены в наш грид, есть стили по умолчанию, которые стоит сбросить или заменить на те, которые указаны в макете. Например, отступ сверху большинству элементов не нужен, а отступ снизу, напротив, сделает вёрстку ритмичной и единообразной. Все подписи к фотографиям сейчас состоят из двух параграфов, но их может стать больше. У последнего отступ снизу уберём.

```
...
.article-img {
  vertical-align: middle;
  max-width: 100%;
  height: auto;
}
```

```
.article-text {
  margin: 0 0 40px 0;
}
```

```
.article-figcaption p {
  margin-top: 0;
  margin-bottom: 10px;
}
```

```
.article-figcaption p:last-child {
  margin-bottom: 0;
}
```

```
.article-figure {
  margin: 0;
}
```

```
.article-figcaption {
  font-size: 16px;
  line-height: 1.5;
}
```

У подписей к изображениям должны быть отступы сверху, но это не касается подписи в конструкции на две колонки.

```
...
.article-figcaption {
  margin-top: 30px;
  font-size: 16px;
  line-height: 1.5;
}
```

```
.article-figure-horizontal {
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 50px;
}
```

```
.article-figure-horizontal .article-figcaption {
  margin-top: 0;
}
```

```
.article-img-wide {
  grid-column: span 3;
}
```

```
.article-figure-wide {
  grid-column: span 2;
}
```

Оформим подвал: его сетку, выравнивание, копирайт. Логотип мы уже стилизовали, когда занимались шапкой.

```
...
<footer class="footer">
  <div class="footer-content container">
    <a class="logo" href="index.html">
      
    </a>
    <p class="footer-copyright">© 2021 Все права защищены</p>
  </div>
</footer>
```

```
...
.article-figure-wide {
  grid-column: span 2;
}
```

/* подвал */

```
.footer-content {
  display: grid;
  grid-template-columns: 145px 1fr;
  grid-column-gap: 50px;
  align-items: center;
```

```
  min-height: 80px;
  box-sizing: border-box;
}
```

```
.footer-copyright {
  margin: 0;
  font-size: 14px;
  line-height: 1.5;
  justify-self: end;
}
```

Осталось добавить отступы до и после основного содержимого, декоративные границы шапки и подвала, а также [сверить получившуюся страницу с Perfect Pixel](#) и подогнать отступы, где это необходимо.

/* подвал */

```
.footer {
  border-top: 1px solid rgba(31,30,28,0.2);
}
```

```
.footer-content {
  ...
}
```

***Дополнительное задание**

Предлагаем для вёрстки макет статьи в блоге о путешествиях — «Traveler's Blog». Это модная статья о путешествиях своим ходом с интересным расположением иллюстраций.

Макет для кейса предоставлен в формате Figma:

<https://www.figma.com/file/f0fBl2plPrr9NPF7shOkcs/travelers?type=design&node-id=0%3A1&mode=design&t=zxkKdGyQu8RJ3Yfm-1>

У сайта есть шапка и подвал, а статья оформлена в несколько колонок. Основной сложностью будет высчитать, сколько колонок и рядов займёт каждый элемент. Рекомендуем оформить заголовок и лид в группу с тегом `<header>`, это упростит задачу. Сетки для каркаса сайта и отдельных элементов нужно построить при помощи *CSS Layout Grid*

Ссылки на дополнительные материалы

1. Про семантическую разметку <https://htmlacademy.ru/blog/html/semantics>
2. Как создать сетку на «гридах» <https://htmlacademy.ru/blog/css/display-grid>
3. Как создавать адаптивные сетки <https://htmlacademy.ru/blog/css/adaptive>