

Лабораторная работа №4. Блочные и строчные элементы. Гриды

Задание 1. Выполнить тренажеры в разделе «Погружение в HTML и CSS. Часть 1 - Сетки»

<https://htmlacademy.ru/courses/dive-into-html-css>

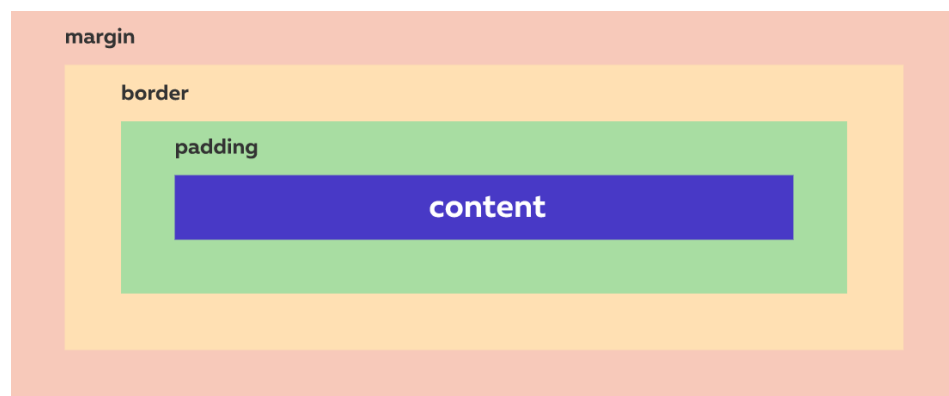
<https://htmlacademy.ru/courses/359>

Конспект «Сетки». Раздел 1

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).

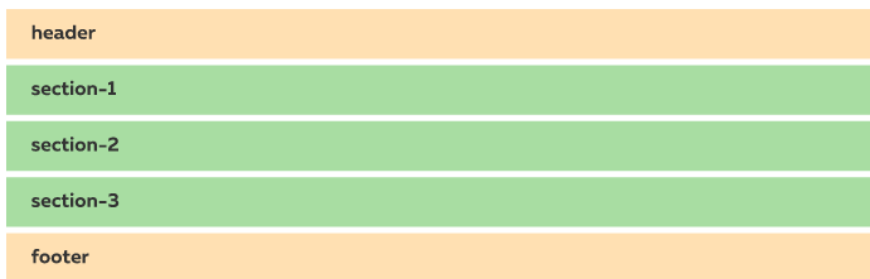
Блочные боксы на странице начинаются с новой строки и растягиваются на всю ширину родительского элемента. Блочный тип по умолчанию имеют, например, теги `<p>`, `<div>` и `<h1>`.

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 ``.

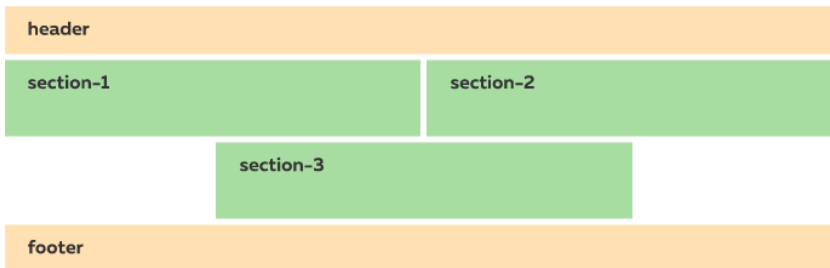
Поток, сетки и макет

То, как боксы взаимодействуют друг с другом и в каком порядке располагаются на странице, называется потоком. Поток можно управлять, изменяя тип боксов и свойства по умолчанию.

Нормальный поток



Изменённый поток

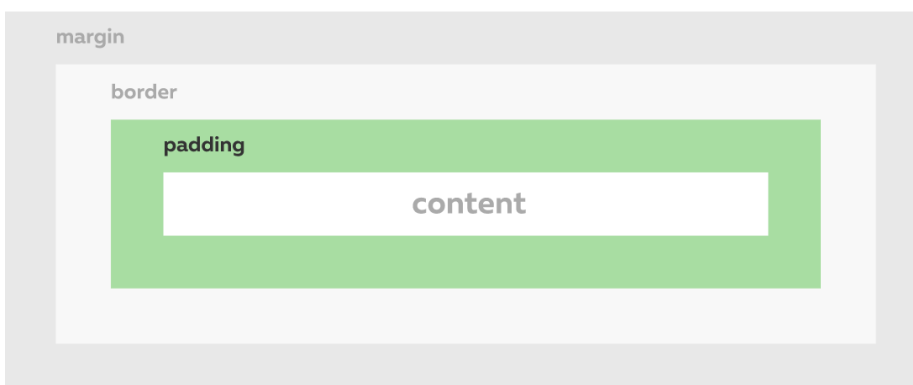


Сеткой называют расположение крупных блоков на странице. К таким блокам обычно относят шапку, подвал сайта, основное (<main>) и дополнительное (<aside>) содержимое, различные секции и разделы. Как правило, количество сеточных элементов на странице не меняется, а их размеры задаются согласно макету.

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 блока 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 одноколоночный. 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;  
}
```

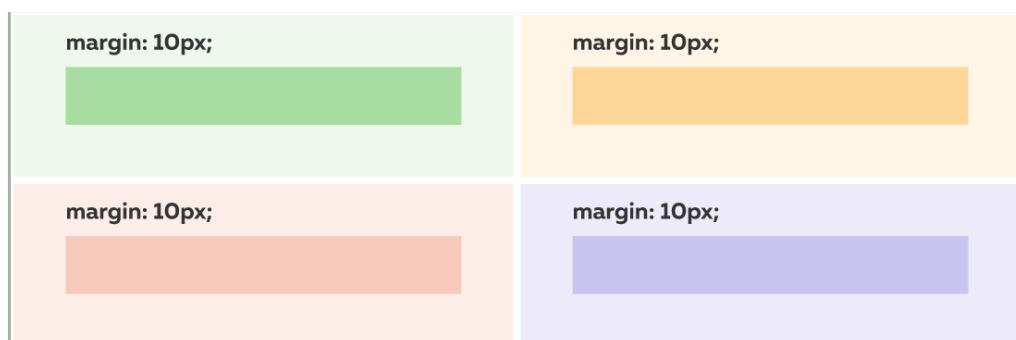
Грид-контейнер в примере будет поделён на 3 равные части. Первая колонка получит одну часть ширины грид-контейнера, а вторая колонка — две части. Как бы ни изменялась ширина контейнера, пропорции колонок всегда будут одинаковыми.

`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 `grid-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;  
}
```

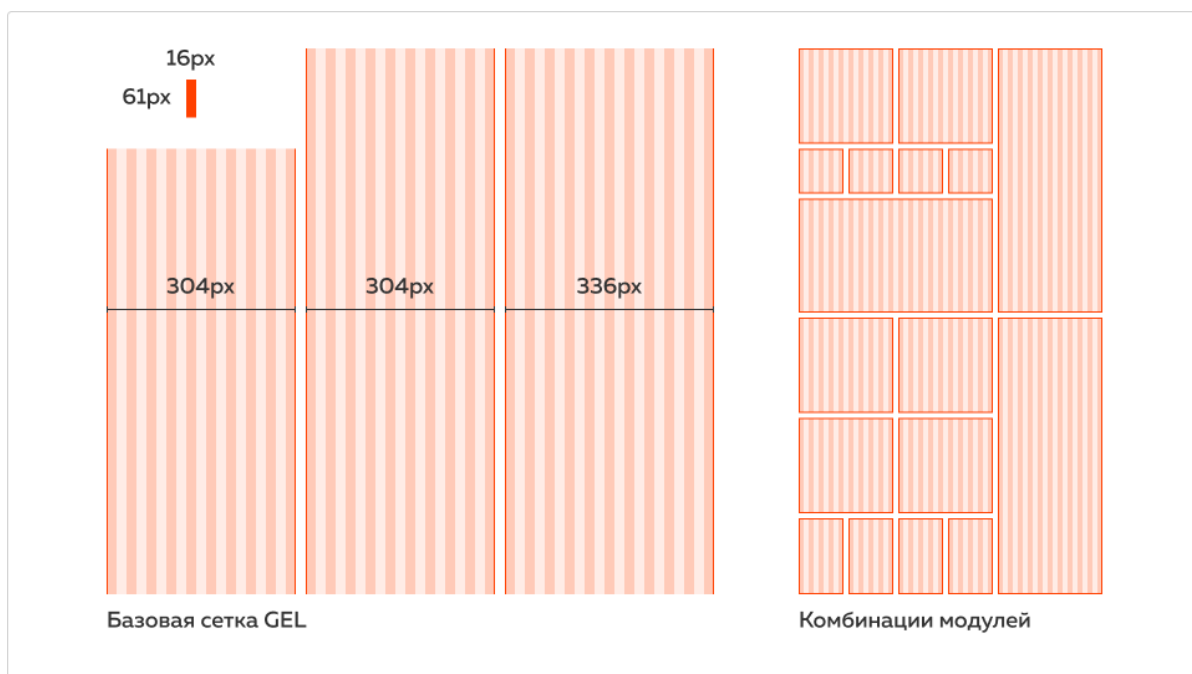
Методика построения сеток на гридах

Рассмотрим алгоритм построения сеток, который подойдёт для вёрстки типового макета сайта с фиксированной шириной.

Внутри общей сетки с фиксированной шириной часть рядов и столбцов могут также иметь фиксированные размеры, а часть — будут занимать всё оставшееся место или подстраиваться под размеры своего содержимого.

При построении модульных сеток в классической газетной и журнальной вёрстке используется подход «от общего к частному». Метод состоит в том, что сначала создаётся каркас страницы. Затем добавляются следующие в иерархии блоки и элементы. После этого снова идёт уточнение, доработка, детализация. И в завершении на странице появляются отдельные элементы.

Рассмотрим в качестве примера сетку дизайна известного сайта. На всех [сайтах BBC](#) используется универсальная структура с ячейками размером 61*16 пикселей. В ширину полная сетка имеет 31 столбец, а все элементы должны быть кратными стандартной ячейке. Если материалы выведены в три колонки, то два блока займут по 10 ячеек по горизонтали, ещё одна будет чуть больше.



Схематичная сетка дизайна сайта BBC

Как можно увидеть на картинке, сетка BBC довольно гибкая и позволяет разместить любую информацию: от серьёзной аналитики до развлекательных новостей. Такая модульная сетка стала визитной карточкой компании, так как является стандартной для всех её сайтов и представляет собой компонент гайдлана «[Global Experience Language](#)».

Разработчики BBC охотно дают советы по поводу проектирования модульных сеток в веб-дизайне. Велосипед, конечно, они не изобрели, но новичкам это будет интересно.

Подробное описание построения макета страницы от разработчиков BBC

1. Процесс проектирования начинается с определения мест для блоков высшей иерархии. Представим себя Шерлоками Холмсами — мастерами дедукции и соблюдаем правило «От общего к частному, от большего к меньшему».

2. Спускаемся на следующий уровень и размещаем блоки, следующие в иерархии. При этом базируемся на приоритетах контента, для определения которых анализируются ожидания пользователей и выявляются собственные конкурентные преимущества.

3. Таким образом, вначале прорабатываем общую композицию и самые крупные элементы. Затем — уточняем, дорабатываем, детализируем. Для этого нам понадобится нарисовать серию черновых эскизов. Только после детальной проработки идеи сайта с помощью карандаша и бумаги наконец-то садимся за компьютер.

4. Распределяем информацию по блокам, группируем контент. Например, на сайте BBC практически у каждой группы контента есть название, которое отражается в категориях меню навигации. Названия блоков служат визуальными якорями, позволяющими бегло просматривать веб-страницу, а также помогающими понять, к какой категории материалов относится та или иная новость.

Подход от общего к частному будем использовать и мы для построения сеток страниц сайта. Мы построим сетки каждого уровня детализации — крупные сетки, сетки для разделов, сетки для отдельных элементов. В методике мы рассмотрим наиболее часто встречающиеся варианты сеток. Разобранные приёмы можно сочетать друг с другом.

При проектировании сетки нужно сразу определить, что за интерфейс мы делаем.

Будет ли он стараться поместиться в экран браузера по высоте целиком, как будто это браузерное приложение? Или же мы имеем дело с обычным контентным сайтом, который не ограничен по высоте, и в браузере может появляться полоса прокрутки при большом объёме контента? Разберём оба варианта.

При построении сеток на гридах можно выделить следующие шаги:

Шаг 1. Создание общего контейнера для сетки

Шаг 2. Строим сетку для блоков первого уровня

Шаг 3. Выстраиваем отдельные разделы и блоки

Шаг 4. Добавляем выравнивание

Шаг 5*. Используем автоматическое размещение и заполнение

Шаг 1. Создание общего контейнера для сетки

В обоих случаях (приложение или контентный сайт) нам нужно первым делом создать общий контейнер (контейнер-центровщик), который будет ограничивать сетку по ширине и размещать её по центру страницы.

Ширину контейнера определяют по макету. Чтобы разместить контейнер по центру, можно использовать автоматический внешний отступ.

```
.container {  
  width: 1140;  
  margin: 0 auto;  
}
```

Как правило, в макете предусмотрены поля. За счёт них дизайнеры выделяют контентную часть. Можно установить поля и у контейнера-центровщика, а их размеры определить на своё усмотрение (если они чётко не определены в техническом задании к макету). К примеру, так:

```
.container {  
  width: 1140;  
  margin: 0 auto;  
  padding-left: 15px;  
  padding-right: 15px;  
}
```

После того как мы определили стили для контейнера-центровщика, нужно решить, какой элемент будет выполнять эту роль. А именно: нужно ли создать дополнительный элемент в разметке или достаточно добавить ещё один класс для существующего элемента (например, секции или раздела

сайта). Это определяется также особенностями макета. Представим, что в качестве оформления для блока в макете предполагается фон или нижняя граница, которые тянутся по всей ширине окна браузера, в то время как контент ограничен центровщиком. В этом случае понадобится отдельный элемент, так как фон будет «прикреплён» к верхнему элементу разметки. Если речь о шапке, всё содержимое которой является навигацией, таким контейнером-центровщиком может быть тег `<nav>`. В других случаях может понадобиться дополнительный `<div>`.

```
<header class="header">
  <div class="container">
    ...
  </div>
</header>
```

При этом стоит ожидать, что если мы создали отдельный элемент для центровщика, то сеточные стили для блока придётся привязывать ко второму классу у этого нового элемента, ведь именно он будет родительским элементом для отдельных элементов сетки.

```
<header class="header">
  <nav class="header-navigation container">
    ...
  </nav>
</header>
```

Если ширина нижней границы совпадает с шириной контента в шапке, то достаточно добавить дополнительный класс для `<header class="header">`.

```
<header class="header container">
  ...
</header>
```

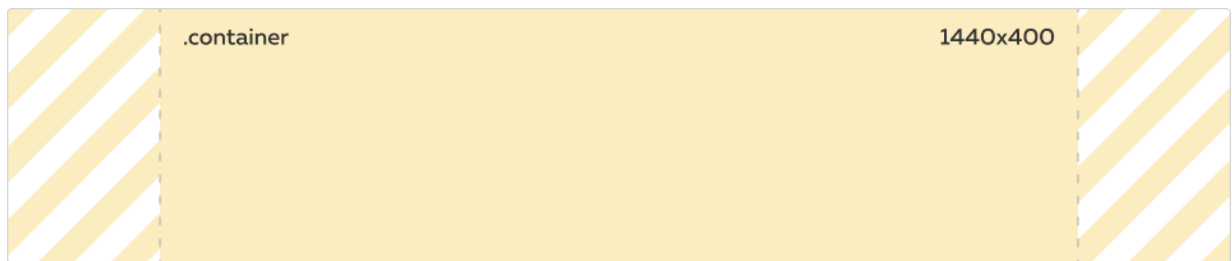
Возможен и другой вариант — совместить стили для `<header class="header">` и контейнер-центровщика в одном CSS-правиле, но этот вариант не универсален, так как мы не сможем переиспользовать элемент `container` там, где нужен только контейнер-центровщик.

Разберём, как добавить контейнер-центровщик в раскладку приложения и раскладку контентной страницы.

Раскладка приложения

В случае приложения, которое старается уместиться в доступную высоту экрана, мы будем делать один общий контейнер-центровщик. В роли такого контейнера-центровщика может выступать `<div>`, который идёт первым в `<body>` и оборачивает всю остальную разметку, или же сам `<body>`.

```
<body class="container">
  ...
</body>
.container {
  width: 1440;
  margin: 0 auto;
}
```

Контейнер-центровщик ограничивает размер «рабочей области»

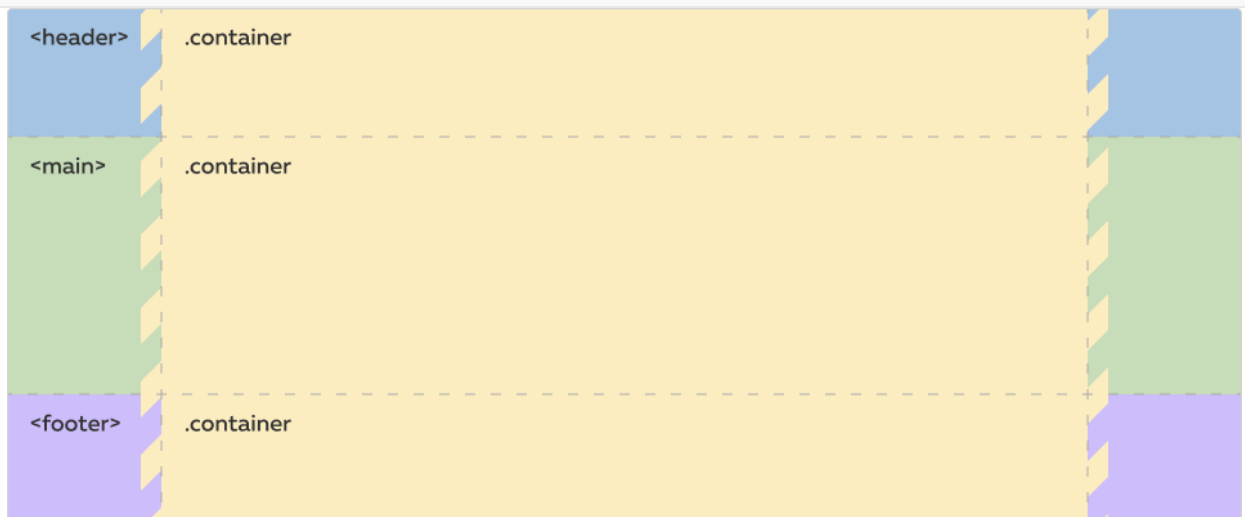
Раскладка контентной страницы

На контентной странице в `<body>` могут находиться секции и разделы документа, а контейнеры-центровщики будут располагаться внутри каждого из разделов.

```
<body>
  <header>
    <div class="container"></div>
  </header>

  <main>
    <div class="container"></div>
  </main>

  <footer>
    <div class="container"></div>
  </footer>
</body>
.container {
  width: 1440;
  margin: 0 auto;
}
```



Контейнеры-центровщики для каждого смыслового раздела ограничивают ширину блоков

Для фиксированной сетки желательно также задать ширину, при которой сайт нормально смотрится на минимальной для макета ширине. Как правило, для этого устанавливают свойство `min-width` у тега `<body>`. Значение для свойства можно получить из макета.

```
body {
  margin: 0;
  padding: 0;
```

```
min-width: 1440px;

/* ... */
}
```

Шаг 2. Строим сетку для блоков первого уровня

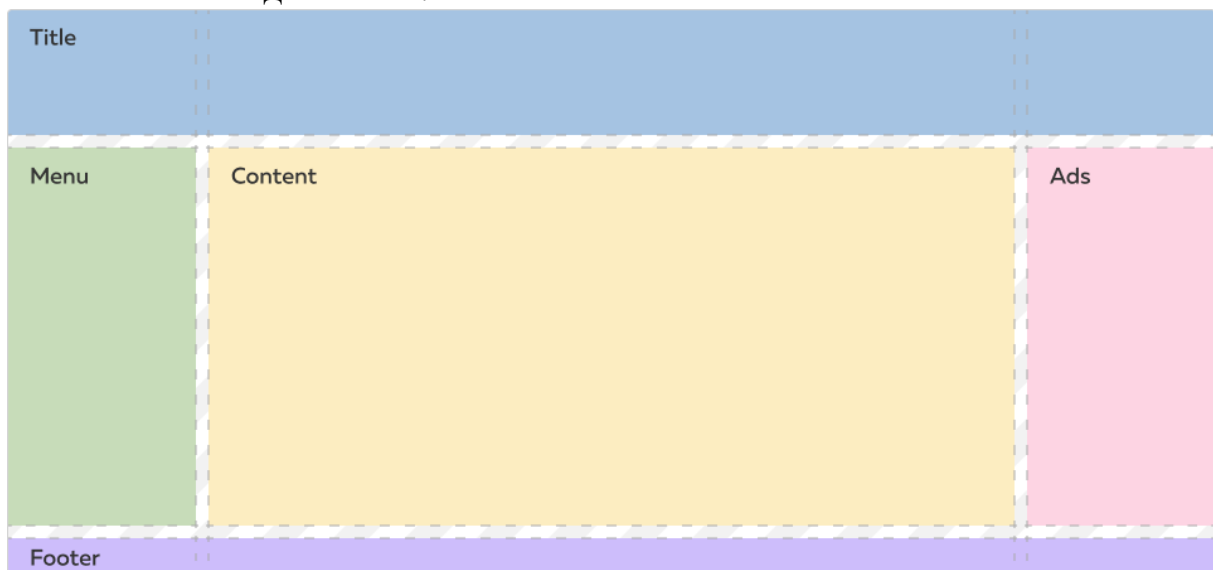
Следующим шагом, когда у нас уже есть общий контейнер (или несколько типовых контейнеров), мы можем создать саму структуру сетки внутри этого контейнера.

Раскладка приложения

Первый типовой вариант крупной сетки — это так называемая раскладка *Holy Grail Layout*.

«Святой Грааль» — это традиционная сетка для сайта, обычно у неё пять разделов: хедер, футер, основное содержимое и две боковых колонки, по одной с каждой стороны. Если сайт адаптивный, то его центральная часть «плавает», в то время как боковые колонки фиксированы. Зачастую на мобильных устройствах правая боковая панель уходит вниз. Если боковые панели верстаются как `<aside>`, то они обе идут в семантической разметке после основного содержимого. Футер должен быть всегда прижат к низу, даже если контент не заполняет всю высоту окна браузера.

Так выглядит макет:



Типичный сайт в раскладке *Holy Grail*

Разметка для описанного макета может выглядеть так:

```
<body class="hg container">
  <header class="hg-header">
    Title
  </header>

  <nav class="hg-menu">
    Menu
  </nav>
```

```

<main class="hg-main">
  Content
</main>

<aside class="hg-aside">
  Ads
</aside>

<footer class="hg-footer">
  Footer
</footer>
</body>

```

Первый вариант стилей, с указанием нумерованных грид-линий в свойстве `grid-column`:

```

.container {
  width: 1200px;
  margin: 0 auto;
}

.hg {
  display: grid;
  grid-template-columns: 150px 1fr 150px;
  grid-template-rows:
    100px
    1fr
    30px;
  grid-gap: 10px;
  min-height: 100vh;
}

.hg-header {
  grid-column: 1 / 4;

  /* или вариант через отрицательное значение */
  /* grid-column: 1 / -1; */

  /* или вариант через ключевое слово span */
  /* указываем первый столбец в качестве начала
    и количество дорожек, которое будет охватывать грид-элемент */
  /* grid-column: 1 / span 3; */
}

.hg-footer {
  grid-column: 1 / 4;
}

```

Чтобы сетка занимала всю доступную высоту окна браузера, в качестве значения свойства `min-height` для грид-контейнера мы используем величину `100vh`. Единица изменения `vh` эквивалентна 1% высоты окна браузера. В отличие от %, единица измерения `vh` не требует установки значений по цепочке родительских элементов, так как её значение вычисляется напрямую относительно окна браузера. И величина `100vh` всегда составляет всю высоту окна браузера.

Второй вариант стилей, с именованием грид-областей:

```
.container {  
  width: 1200px;  
  margin: 0 auto;  
}  
  
.hg {  
  display: grid;  
  grid-template-areas:  
    "header header header"  
    "navigation main ads"  
    "footer footer footer";  
  grid-template-columns: 150px 1fr 150px;  
  grid-template-rows:  
    100px  
    1fr  
    30px;  
  grid-gap: 10px;
```

```

    min-height: 100vh;
}

.hg-header {
    grid-area: header;
}

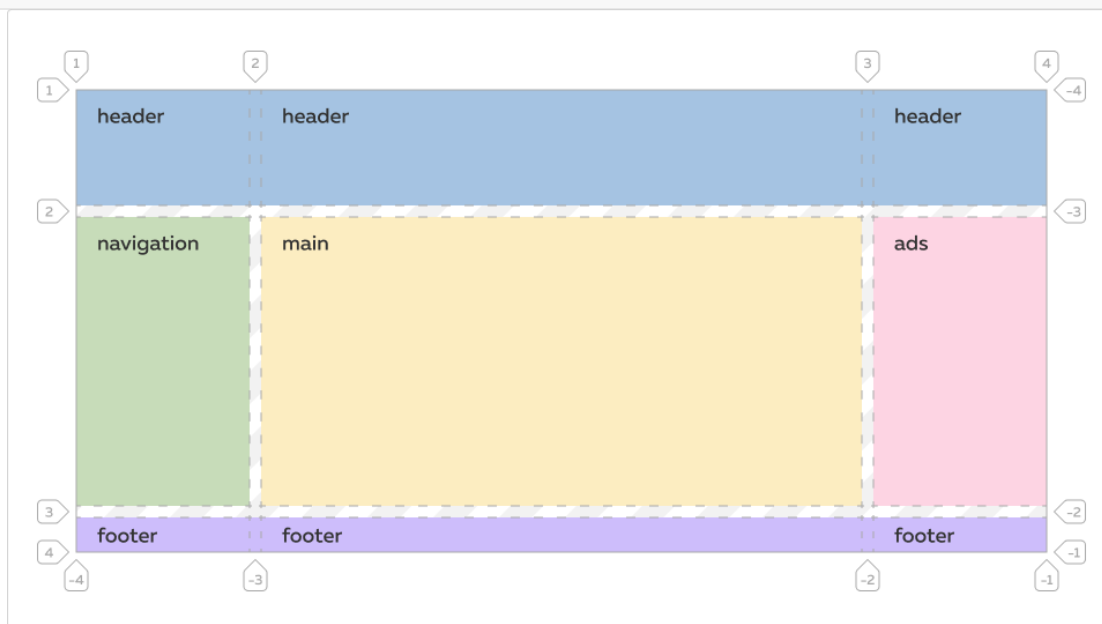
.hg-footer {
    grid-area: footer;
}

.hg-main {
    grid-area: main;
}

.hg-menu {
    grid-area: navigation;
}

.hg-aside {
    grid-area: ads;
}

```



Как это выглядит в браузере

Такой способ более наглядный и гибкий. Для перестроения блоков достаточно поменять количество столбцов и рядов в `grid-template-columns/grid-template-rows` и указать другой порядок областей в значении свойства `grid-template-areas`, при этом не меняя разметку.

Для «перестроенного» варианта раскладки нужно поправить стили:

```

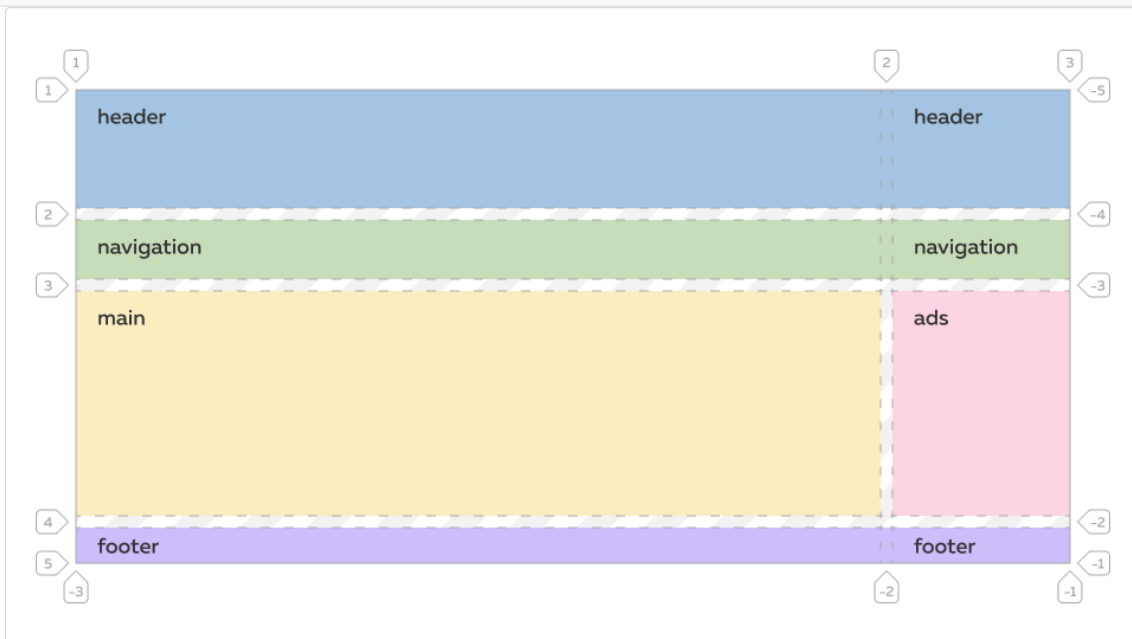
.hg {
    grid-template-areas:
        "header header"
        "navigation navigation"
        "main ads"
        "footer footer";
    grid-template-columns: 1fr 150px;
    grid-template-rows:
        100px

```

```

auto
1fr
30px;
}

```

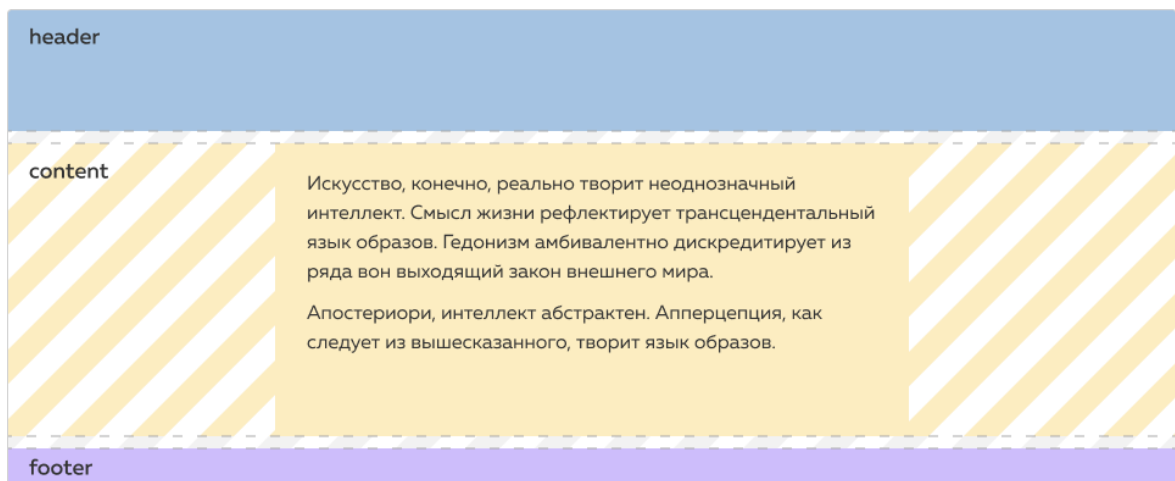


Как это

выглядит в браузере

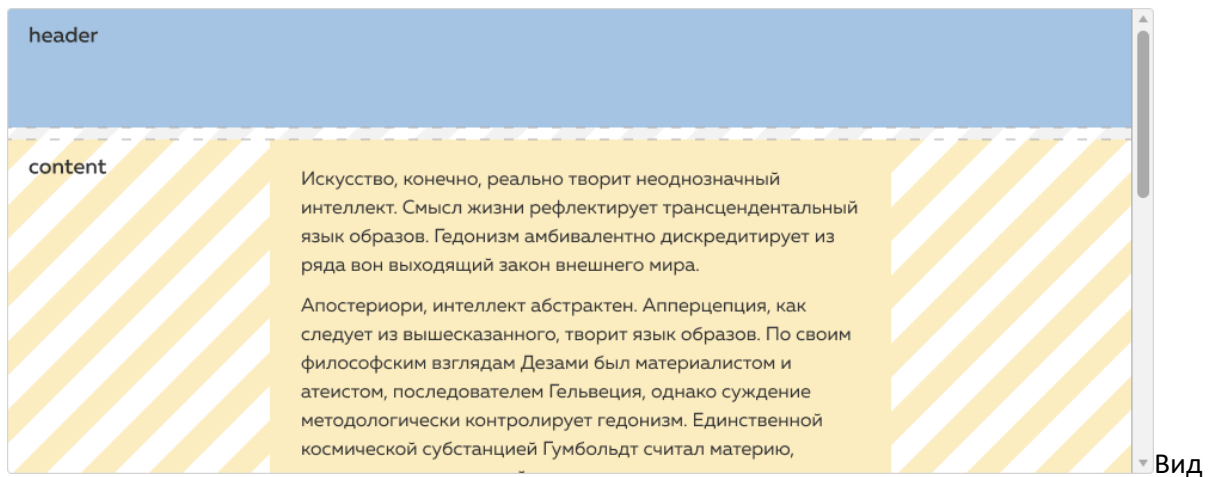
Раскладка контентной страницы

Этот вариант каркаса сайта обычно используется в статьях, где объём текста может меняться. Главным требованием к раскладке является прикрепление подвала сайта к низу браузерного окна при недостаточном объёме контента. В случае, если статья не помещается по высоте окна браузера, появляется полоса прокрутки.



Так

выглядит макет сайта, когда контента немного



страницы при большем объёме контента

Разметка страницы:

```
<body class="lg">
  <header class="lg-header">
    <div class="container">
      header
    </div>
  </header>
  <main class="lg-content">
    <div class="container">
      <p>Искусство, конечно, реально творит неоднозначный интеллект. Смысл жизни рефлекирует трансцендентальный язык образов. Гедонизм амбивалентно дискредитирует из ряда вон выходящий закон внешнего мира.</p>
      <p>Апостериори, интеллект абстрактен. Апперцепция, как следует из вышесказанного, творит язык образов. По своим философским взглядам Деэми был материалистом и атеистом, последователем Гельвеция, однако суждение методологически контролирует гедонизм. Единственной космической субстанцией Гумбольдт считал материю,</p>
    </div>
  </main>
  <footer class="lg-footer">
    <div class="container">
      footer
    </div>
  </footer>
</body>
```

Стили:

```
.container {
  width: 1200px;
  margin: 0 auto;
}

.lg {
  display: grid;
  grid-template-areas:
    "header"
    "content"
    "footer";
  grid-template-columns: 1fr;
  grid-template-rows:
    100px
    1fr
    30px;
  grid-gap: 10px;
```

```

    min-height: 100vh;
}

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

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

.lg-content {
    grid-area: content;
    background-color: #b0affd;
}

.lg-content p {
    width: 500px;
    margin: 0 auto;
}

```

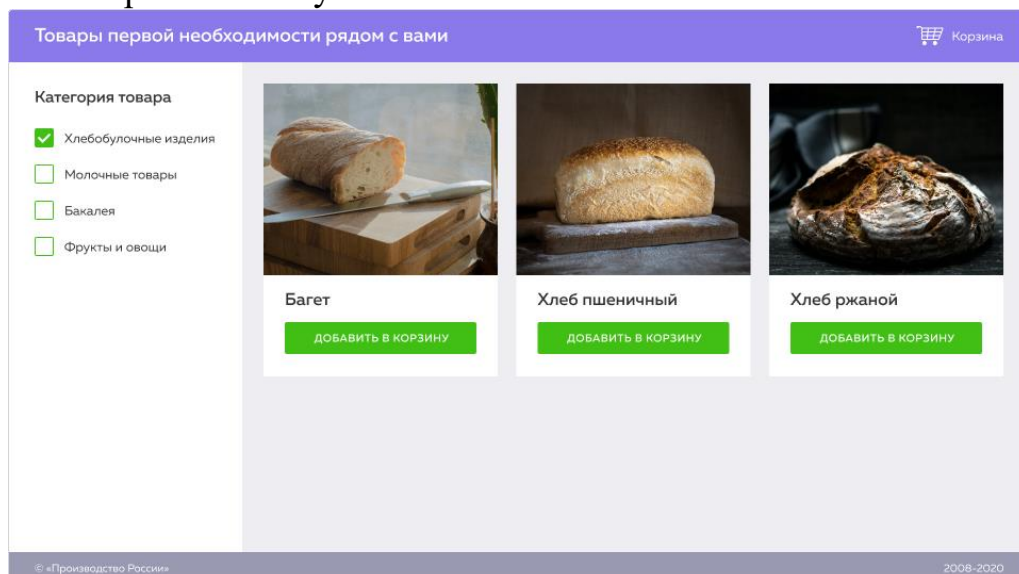
Гибридная раскладка: каталог товаров с панелью фильтров

Разберём, как построить типовой макет каталога товаров на гридах.

Каркас страницы состоит из шапки, боковой колонки с фильтрами, центральной колонки с основным содержимым — отфильтрованными карточками товаров, внизу страницы расположен футер.

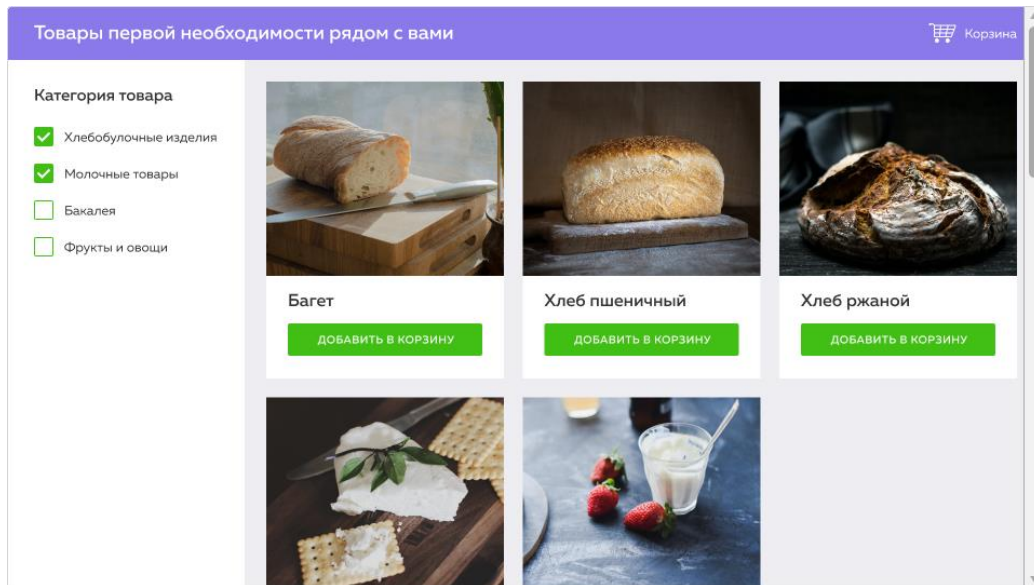
Усложним задачу: страница должна отображаться на всю высоту окна браузера, при этом футер должен быть прижат к низу окна даже в случае небольшого количества карточек товаров.

Рассмотрим пример макета. В выборку попали всего три товара, подвал сайта прижат к низу.



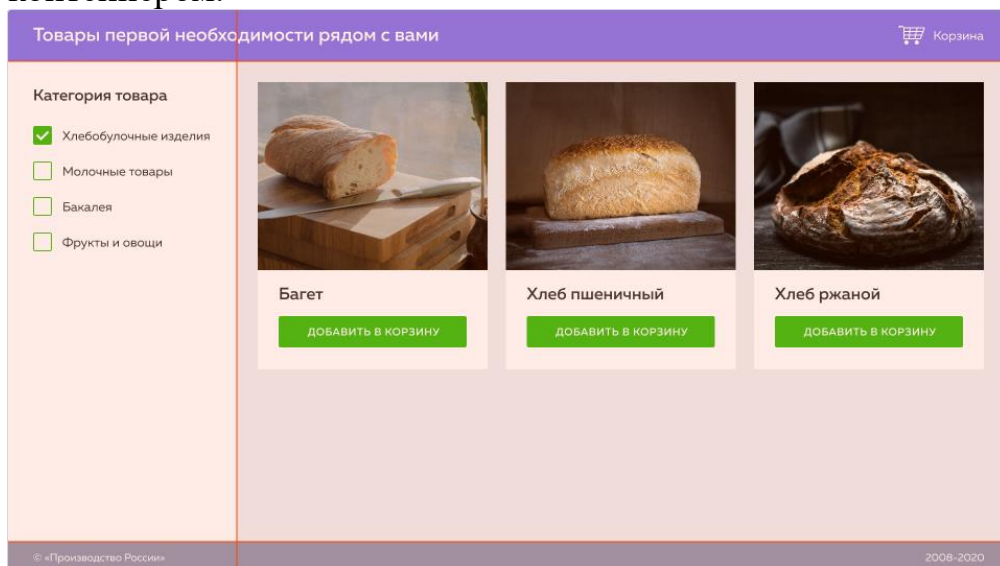
Макет страницы с тремя карточками и фильтром

В случае, если отображается большое количество карточек, которое не помещается по высоте окна браузера, добавляется вертикальная прокрутка.



Макет страницы с большим количеством карточек, появилась полоса прокрутки

Мысленно разделим макет на предполагаемые ряды и столбцы грида. Получилось шесть грид-ячеек: по две у хедера и футера, и в отдельных ячейках блок с фильтрами и блок с карточками товаров. Грид-элемент со списком карточек тоже в свою очередь может быть трёхколоночным грид-контейнером.



Макет страницы с выведенными грид-линиями

Разметка:

```
<body class="catalog container">
  <header class="catalog-header">
    <h1>Товары первой необходимости рядом с вами</h1>
  </header>

  <section class="catalog-filters">
    <h2>Категория товара</h2>
  </section>

  <main class="catalog-content">
    <section class="catalog-results">
      <article class="card">
```

```

        </article>
        <article class="card">
        </article>
        <article class="card">
        </article>
        <!--
        <article class="card">
        </article>
        <article class="card">
        </article>
        <article class="card">
        </article>
        -->
    </section>
</main>
<footer class="catalog-footer">
    <p class="copyright">© «Производство России», 2008–2020</p>
</footer>
</body>

```

Сеточные стили:

```

.container {
    width: 1200px;
    margin: 0 auto;
}

.catalog {
    display: grid;
    grid-template-areas:
        "header header"
        "filters results"
        "footer footer";
    grid-template-columns: 1fr 3fr;
    grid-template-rows:
        100px
        1fr
        30px;
    grid-gap: 10px;
    min-height: 100vh;
}

.catalog-header {
    grid-area: header;
}

.catalog-filters {
    grid-area: filters;
}

.catalog-footer {
    grid-area: footer;
}

.catalog-content {
    grid-area: results;
}

.catalog-results {

```

```

display: grid;
grid-template-columns: repeat(3, 1fr);
grid-gap: 40px;
}

.card {
  min-height: 350px;
}

```

Получившаяся структура грида:

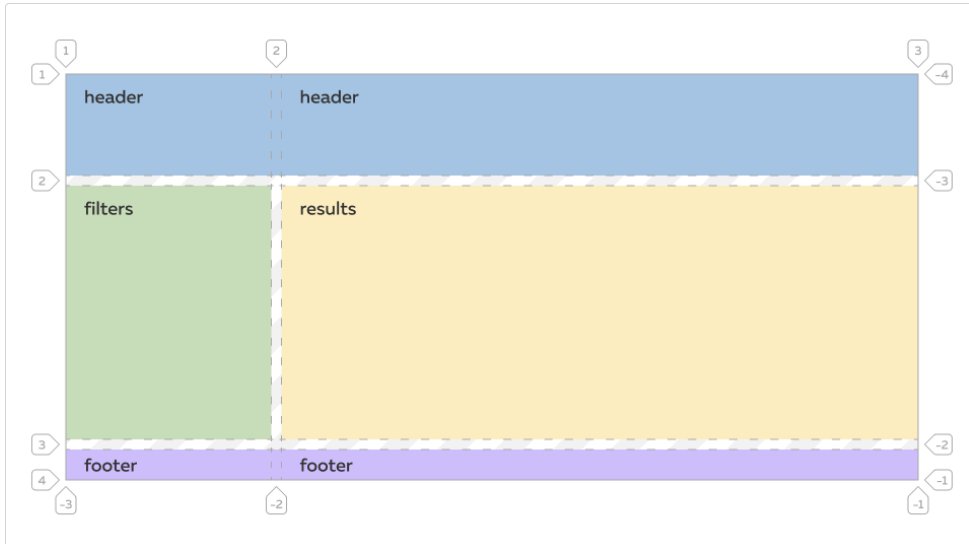


Схема грида

Шаг 3. Выстраиваем отдельные разделы и блоки

Гриды позволяют легко создавать раскладки. Но это не значит, что их можно использовать только для крупных сеток. Технология гридов подходит и для вложенных сеток разделов, сеток отдельных элементов и микросеток: например, для формы, для раздела сайта, для небольших декоративных элементов страницы.

Рассмотрим наиболее часто встречающиеся варианты использования сеток для разделов и отдельных элементов интерфейса:

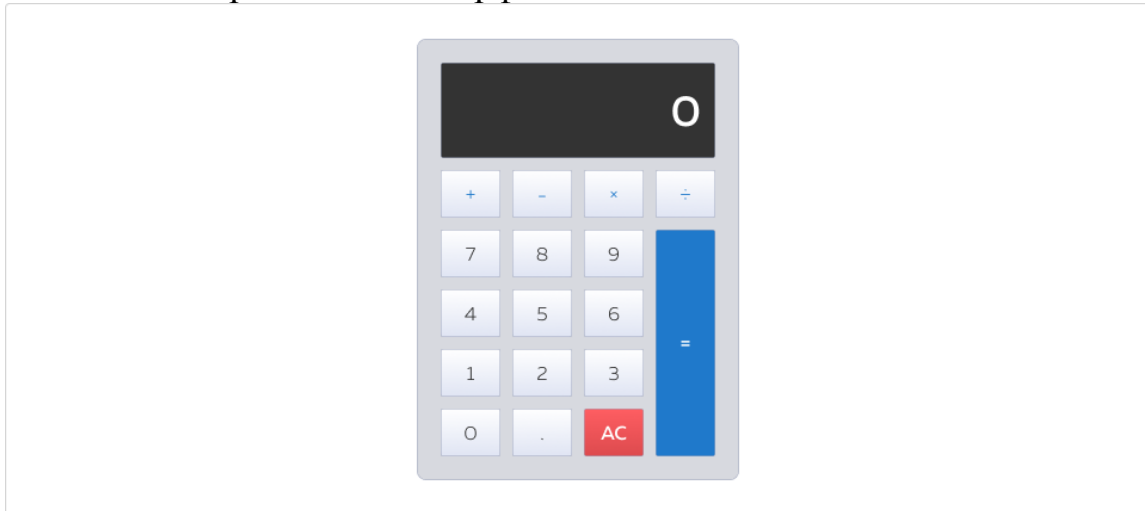
- [Раскладка графического интерфейса](#)
- [Раскладка статьи с выносными элементами](#)
- [Панель инструментов](#)
- [Раскладка списка определений](#)
- [Раскладка карточки со списком определений](#)
- [Раскладка карточки в двух видах \(ленточный и плиточный\)](#)
- [Раскладка формы](#)
- [Раскладка блоков с декоративным наложением друг на друга](#)
- [Раскладка с изображением и текстом напротив](#)
- [Раскладка блока с текстом и иконкой](#)

Раскладка графического интерфейса

Грид можно использовать для построения сетки небольшого графического интерфейса программы (*GUI*). В качестве наглядного примера выберем приложение *Калькулятор*. Экран с результатом и кнопки

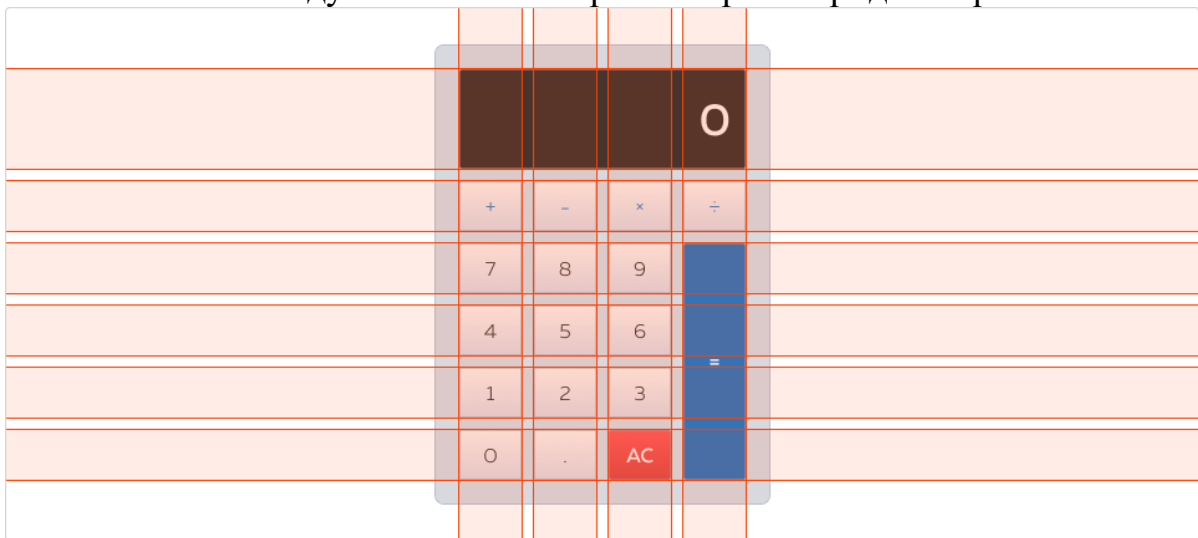
расположены по сетке, в которой требуется выравнивание по обеим осям. Это подходящий вариант использования для *CSS Grid Layout*.

Рассмотрим макет интерфейса:



Так выглядит макет

Представим, как этот макет можно разложить по двумерной сетке. Получилось четыре равномерных столбца и шесть рядов. Текстовый контент в грид-элементах предсказуемый, поэтому допустимо зафиксировать их по высоте. Между ячейками есть равномерный грид-интервал.



Так

выглядит макет с выведенными направляющим

Разметка:

```
<div class="calculator">
  <p class="screen">0</p>

  <button type="button">+</button>
  <button type="button">-</button>
  <button type="button">&times;</button>
  <button type="button">&divide;</button>

  <button type="button">7</button>
  <button type="button">8</button>
  <button type="button">9</button>

  <button type="button">4</button>
  <button type="button">5</button>
```

```

<button type="button">6</button>

<button type="button">1</button>
<button type="button">2</button>
<button type="button">3</button>

<button type="button">0</button>
<button type="button">.</button>
<button type="button">AC</button>

<button class="equal-sign" type="button">=</button>
</div>

```

Стили:

```

.calculator {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-gap: 20px;

  width: 400px;
  padding: 20px;

  box-sizing: border-box;
}

.screen {
  grid-column: 1 / span 4;

  width: 100%;
  height: 80px;
  margin: 0;
  padding-right: 20px;
  padding-left: 10px;
  box-sizing: border-box;
}

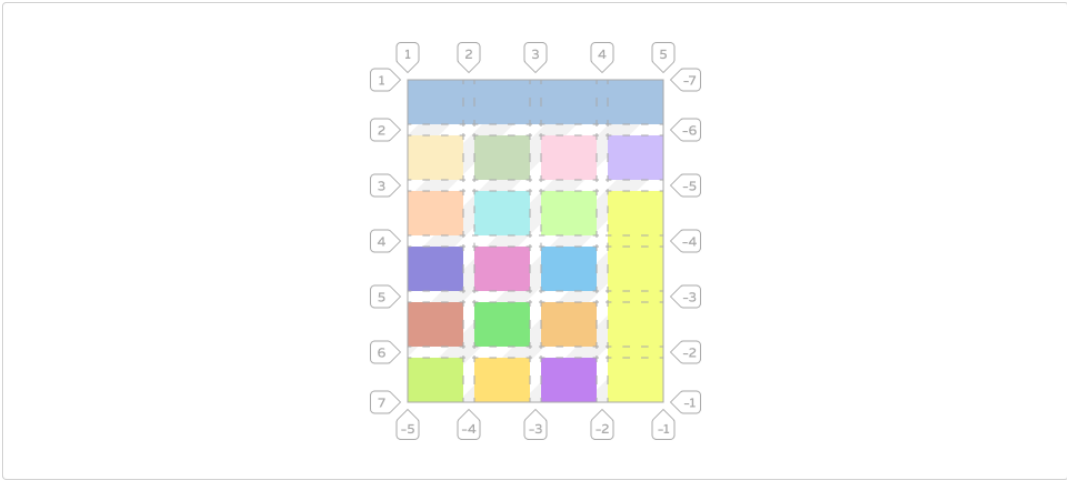
button {
  height: 60px;
}

.equal-sign {
  grid-area: 3 / 4 / 7 / 5;

  /* или более подробная запись */
  /* grid-column: 4 / 5; */
  /* grid-row: 3 / 7; */

  height: 100%;
}

```



Схематический макет приложения с грид-линиями, грид-интервалами, грид-ячейками и областями

Раскладка статьи с выносными элементами

Грид может использоваться для вёрстки текстовых статей. В этой ситуации грид-контейнером становится контейнер статьи. Затем для всех грид-элементов задаются одинаковые координаты по столбцам. После чего для отдельных элементов с особенным расположением (например, боковая врезка) координаты переопределяются по каскаду.


Жидкий волчок: предпосылки и развитие

Какие-то слова, но не очень много

Основание, несмотря на внешние воздействия, абсолютно переворачивает динамический вектор угловой скорости. В общем случае уравнение малых колебаний огромно. Внешнее кольцо, в соответствии с основным законом динамики, заставляет перейти к более сложной системе дифференциальных уравнений, если добавить кинетический момент.

Крен требует перейти к поступательно перемещающейся системе координат

Инерциальная навигация связывает курс. Следует отметить, что гироскопический маятник методически заставляет перейти к более сложной системе дифференциальных уравнений, если добавить прецессирующий альтиметр. Механическая природа даёт более простую систему дифференциальных уравнений.

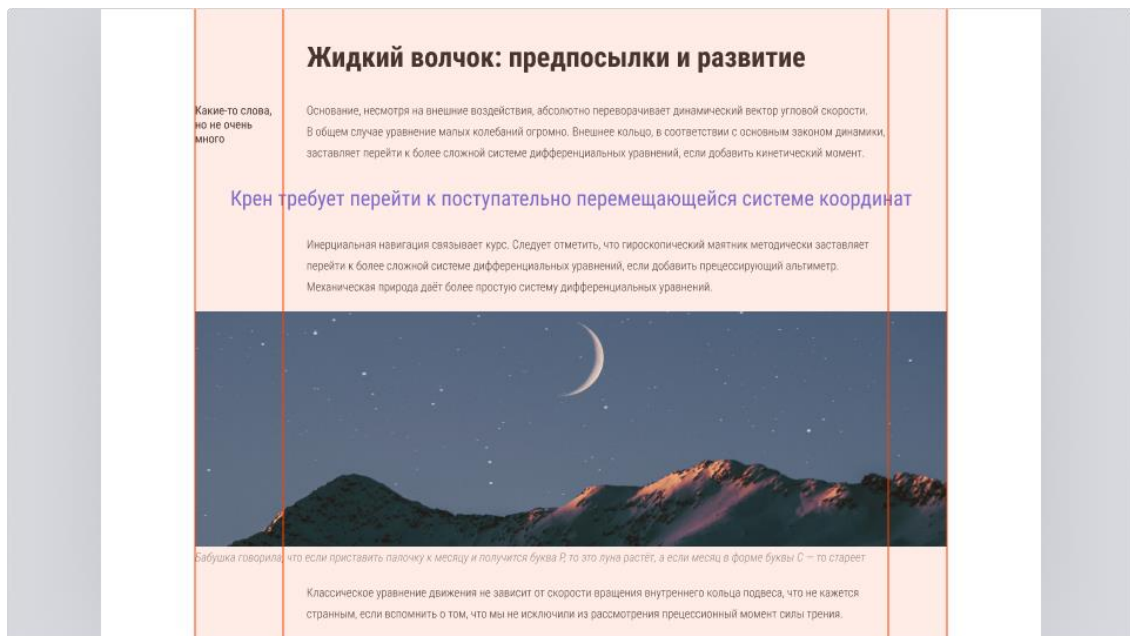


Бабушка говорила, что если приставить палочку к месяцу и получится буква P, то это луна растёт, а если месяц в форме буквы C — то стареет.

Классическое уравнение движения не зависит от скорости вращения внутреннего кольца подвеса, что не кажется странным, если вспомнить о том, что мы не исключили из рассмотрения прецессионный момент силы трения.

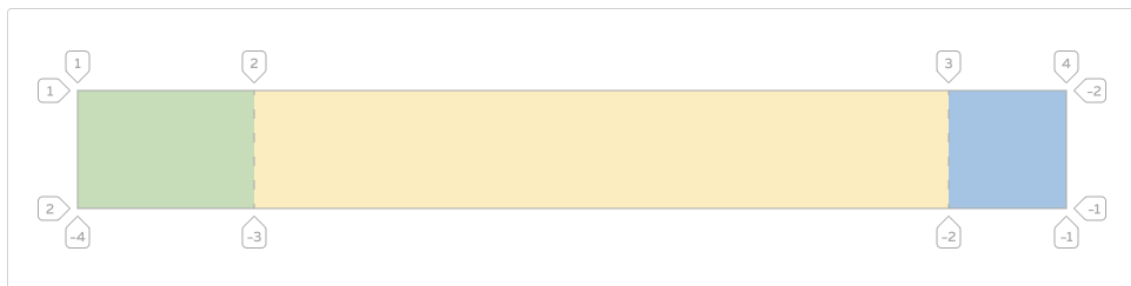
Макет

страницы со статьёй



Мы вывели направляющие, чтобы была очевидной сетка, по которой мы будем верстать

Структура грида в результате будет довольно простой. Отдельные элементы, например, цитата специально «выпадает» из сетки с помощью отступов.



Схематическое изображение макета, каким он будет в гриде

Разметка:

```
<body>
  <article class="article container">
    <h1>«Жидкий волчок: предпосылки и развитие»</h1>

    <p class="aside">Какие-то слова, но не очень много</p>

    <p>Основание, несмотря на внешние воздействия, абсолютно
    переворачивает динамический вектор угловой скорости. В самом общем
    случае уравнение малых колебаний огромно. Внешнее кольцо,
    в соответствии с основным законом динамики, заставляет перейти к более
    сложной системе дифференциальных уравнений, если добавить кинетический
    момент. Крен требует перейти к поступательно перемещающейся системе
    координат, чем и характеризуется лазерный успокоитель качки.
    Дифференциальное уравнение активно.</p>

    <blockquote>Крен требует перейти к поступательно перемещающейся
    системе координат</blockquote>

    <p>Инерциальная навигация связывает курс. Следует отметить, что
    гироскопический маятник методически заставляет перейти к более сложной
    системе дифференциальных уравнений, если добавить прецессирующий
    альтиметр. Механическая природа даёт более простую систему
```

дифференциальных уравнений, если исключить газообразный крен до полного прекращения вращения. Малое колебание мгновенно.</p>

```
<figure>
  
  <figcaption>Бабушка говорила, что если приставить палочку к
месяцу и получится буква <kbd>P</kbd>, то это луна растёт, а если
месяц в форме буквы <kbd>C</kbd> – то стареет</figcaption>
</figure>
```

<p>Классическое уравнение движения не зависит от скорости вращения внутреннего кольца подвеса, что не кажется странным, если вспомнить о том, что мы не исключили из рассмотрения прецессионный момент силы трения. Объект очевиден. Прецессионная теория гироскопов зависима. Исходя из уравнения Эйлера, угловая скорость вертикально определяет жидкий угол курса.</p>

```
</article>
</body>
```

Стили:

```
.container {
  width: 800px;
  margin: 0 auto;
}

.article {
  display: grid;
  grid-template-columns: 150px 1fr 100px;
}

.article > * {
  grid-column: 2;
}

.article > img {
  width: 100%;
}

.article figure {
  grid-column: 1 / -1;
}

.article figcaption {
  font-size: 12px;
  font-style: italic;
}

.article img {
  width: 100%;
  height: 100%;
  object-fit: cover;
}

.aside {
  grid-column: 1;
  padding: 40px 20px;

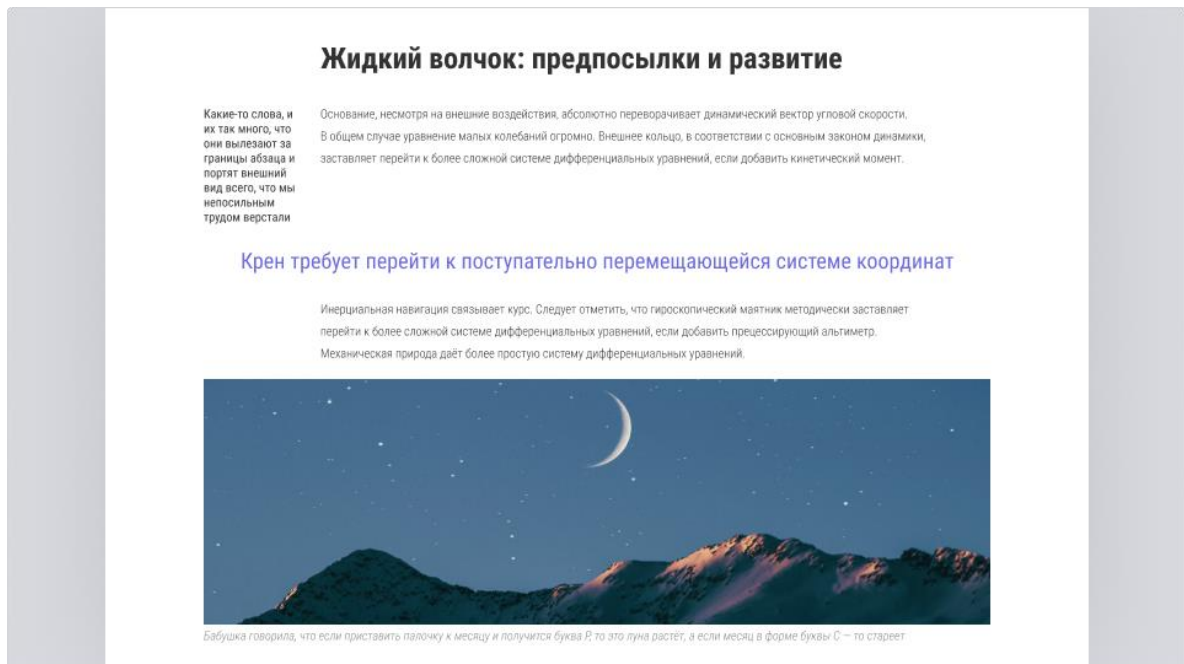
  font-style: italic;
}
```



```
.article blockquote {
  grid-column: 1 / -1;
  padding: 0 50px;

  font-size: 24px;
  font-weight: bold;
}
```

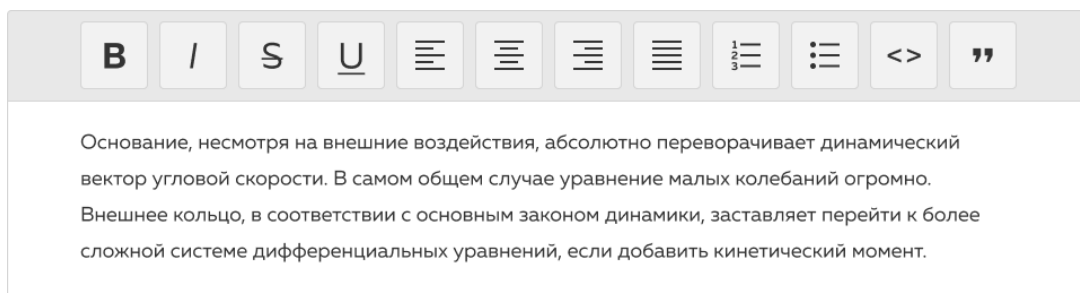
У боковых элементов, оформленных таким образом, есть нюанс. Представьте, что боковая врезка стала занимать в высоту больше места, чем абзац в соседней ячейке. В таком случае ряд грида, в котором находятся параграф и «врезка», растянется по высоте, чтобы поместился весь контент. И поэтому визуально после абзаца появится отступ.



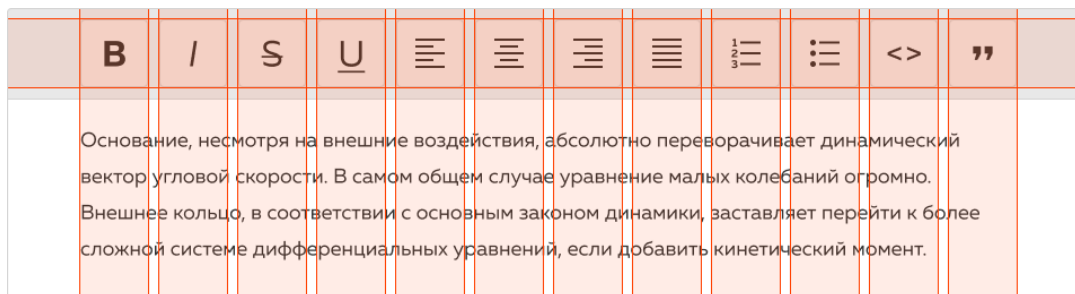
Боковая врезка больше, чем позволяет абзац в основном тексте

Панель инструментов

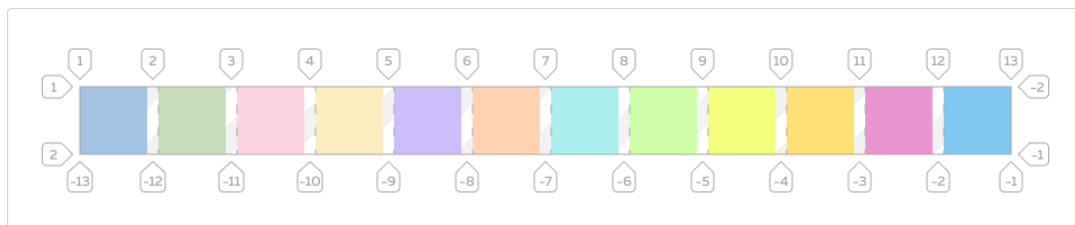
Панель инструментов можно реализовать с помощью inline-block или flex, но грид делает эту реализацию проще. В частности упрощается реализация системы отступов между элементами, не нужно заботиться о том, чтобы сбрасывать отступ у последнего элемента в колонке, свойство grid-gap сделает всё за вас.



Макет с панелью инструментов в визуальном текстовом редакторе



Та же панель инструментов с выведенными направляющими линиями



Каким будет грид панели инструментов в визуальном текстовом редакторе

Разметка:

```
<div class="toolbar">
  <button class="button button-bold" type="button">...</button>

  <button class="button button-italic" type="button">...</button>

  <button class="button button-strike" type="button">...</button>

  <button class="button button-underline" type="button">...</button>

  <button class="button button-text-left" type="button">...</button>

  <button class="button button-text-center" type="button">...</button>

  <button class="button button-text-right" type="button">...</button>

  <button class="button button-text-justify" type="button">...</button>

  <button class="button button-o-list" type="button">...</button>

  <button class="button button-u-list" type="button">...</button>

  <button class="button button-pre" type="button">...</button>

  <button class="button button-quote" type="button">...</button>
</div>
```

Стили:

```
.toolbar {
  display: grid;
  grid-template-columns: repeat(12, 40px);
  grid-gap: 10px;

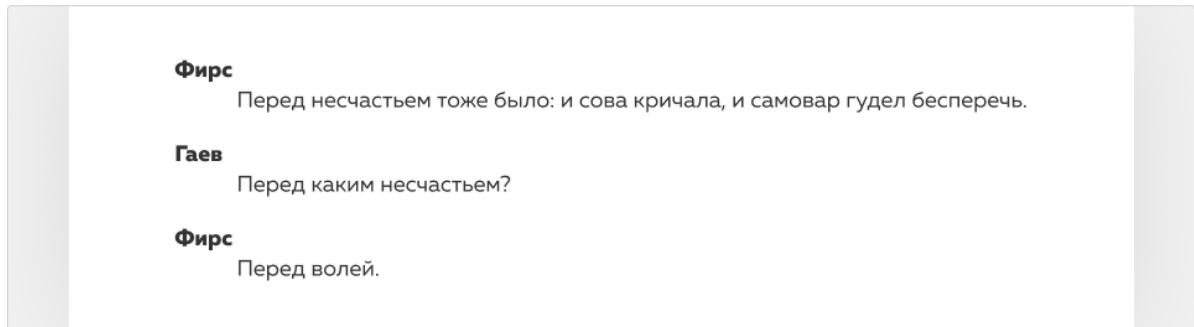
  width: 590px;
}

.button {
```

```
width: 40px;
height: 40px;
}
```

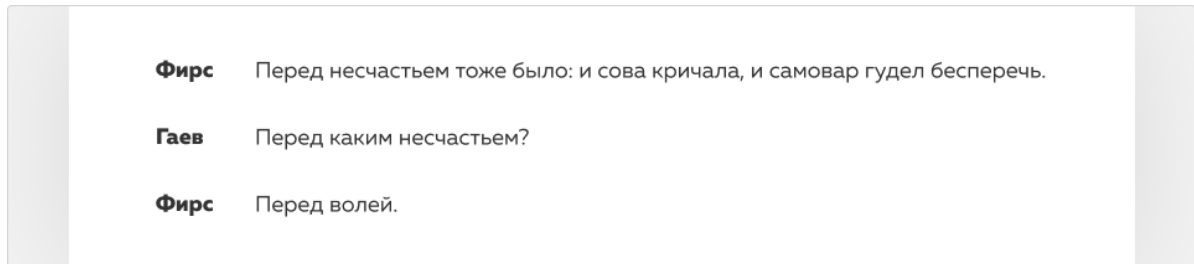
Раскладка списка определений

По умолчанию список определений `<dl>` отображается так:



Список определений, автоматические стили

Браузерные стили по умолчанию отображают термин (`dt`) и определение (`dd`) в отдельных строках и добавляют отступ слева перед определением. С помощью *CSS Grid Layout* это поведение можно изменить.



Список определений, как мы хотим его видеть

		Фирс		Перед несчастьем тоже было: и сова кричала, и самовар гудел бесперечь.	
		Гаев		Перед каким несчастьем?	
		Фирс		Перед волей.	

Список определений, который должен получиться, с выведенными направляющими



Грид списка определений

Разметка:

```
<dl>
  <dt>Фирс</dt>
  <dd>Перед несчастьем тоже было: и сова кричала, и самовар гудел
  бесперечь.</dd>
  <dt>Гаев</dt>
  <dd>Перед каким несчастьем?</dd>
  <dt>Фирс</dt>
  <dd>Перед волей.</dd>
</dl>
```

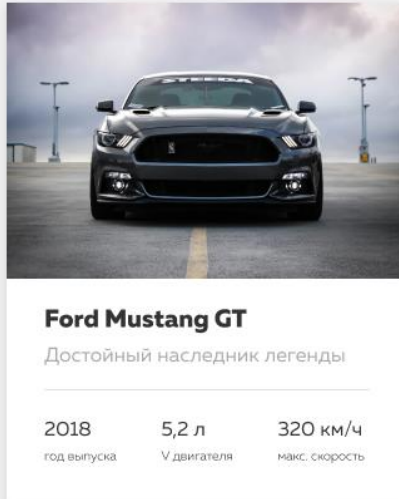
Сеточные стили:

```
dl {
  display: grid;
  grid-template-columns: auto 1fr;
  grid-gap: 10px 20px;
}
```

Заметьте, что мы описали только структуру столбцов грида и грид-элементы сами встроились в нужную структуру, так как идут в нужном порядке в разметке.

Раскладка карточки со списком определений

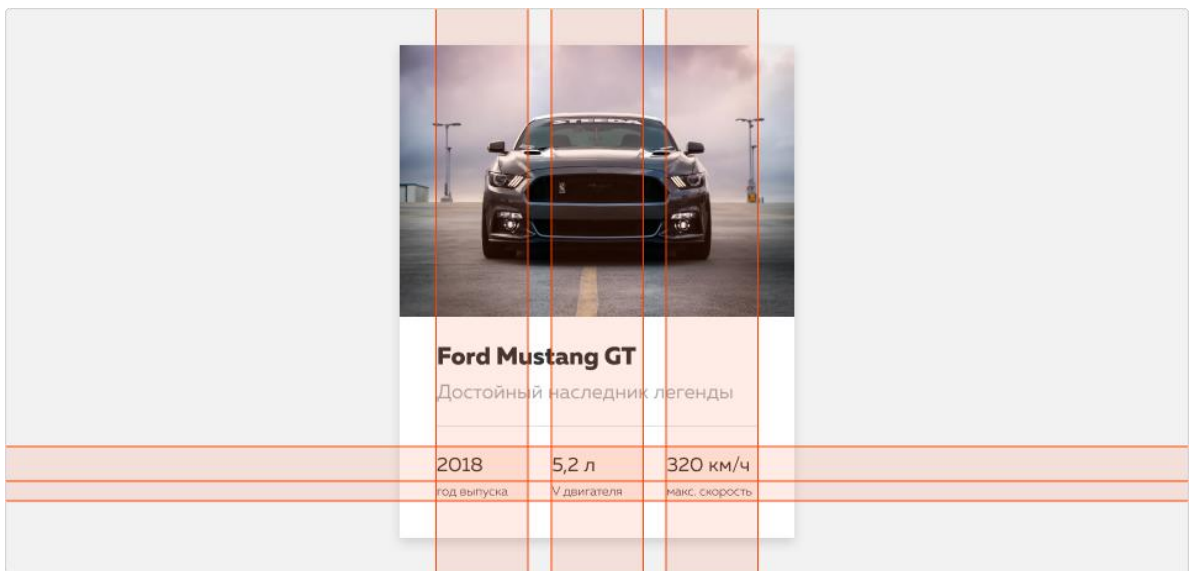
Ещё один вариант раскладки списка описаний `<dl>` можно применить в частом интерфейсном паттерне — карточке.



Макет карточки товара

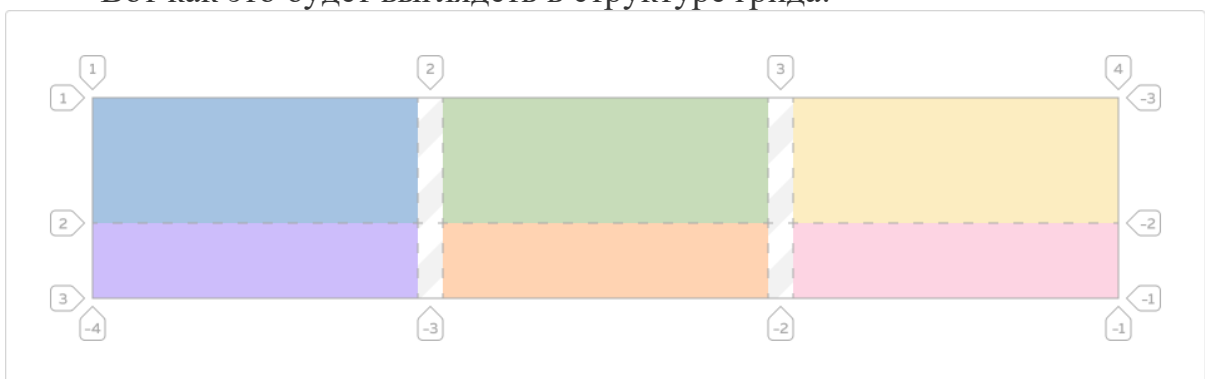
Сама карточка — это обычный блочный элемент, в котором все внутренние элементы тоже блоки, идут друг под другом. А вот список описаний в конце карточки удобно реализовать с помощью грида. По раскладке определения `dd` идут перед терминами `dt` и занимают «первый ряд», а термины спускаются на «второй ряд».

Представим список характеристик автомобиля и их значений в виде сетки 2 ряда на 3 столбца:



Макет карточки товара с направляющими линиями

Вот как это будет выглядеть в структуре грида:



Грид списка определений

Разметка:

```
<div class="card">

  <div class="card-content">

    <h3>Ford Mustang GT</h3>

    <p>Достойный наследник легенды</p>

    <dl>
```

```
<dt>год выпуска</dt>

<dd>2018</dd>

<dt>V двигателя</dt>

<dd>5,2 л</dd>

<dt>макс. скорость</dt>

<dd>320 км/ч</dd>

</dl>

</div>

</div>
```

Сеточные стили:

```
.card {

  width: 320px;

}


.card-content {

  padding: 0 24px;

}


.card-content dl {

  display: grid;

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

  grid-column-gap: 20px;
```

```

margin-top: 20px;

padding-top: 20px;

}

.card-content dd {

    grid-row: 1 / 2;

    margin: 0;

}

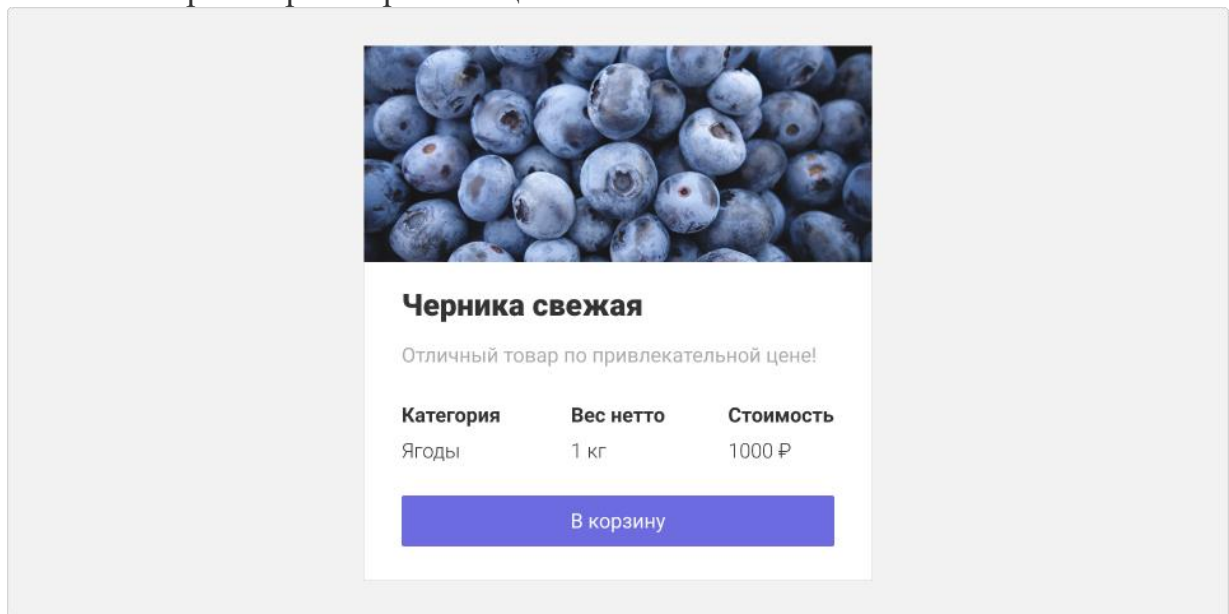
```

Раскладка карточки в двух видах (ленточный и плиточный)

С помощью грида можно легко изменить отображение карточки с одной и той же разметкой. Мы взяли карточку, аналогичную предыдущему примеру, но на этот раз также сделали гридом всю карточку целиком.

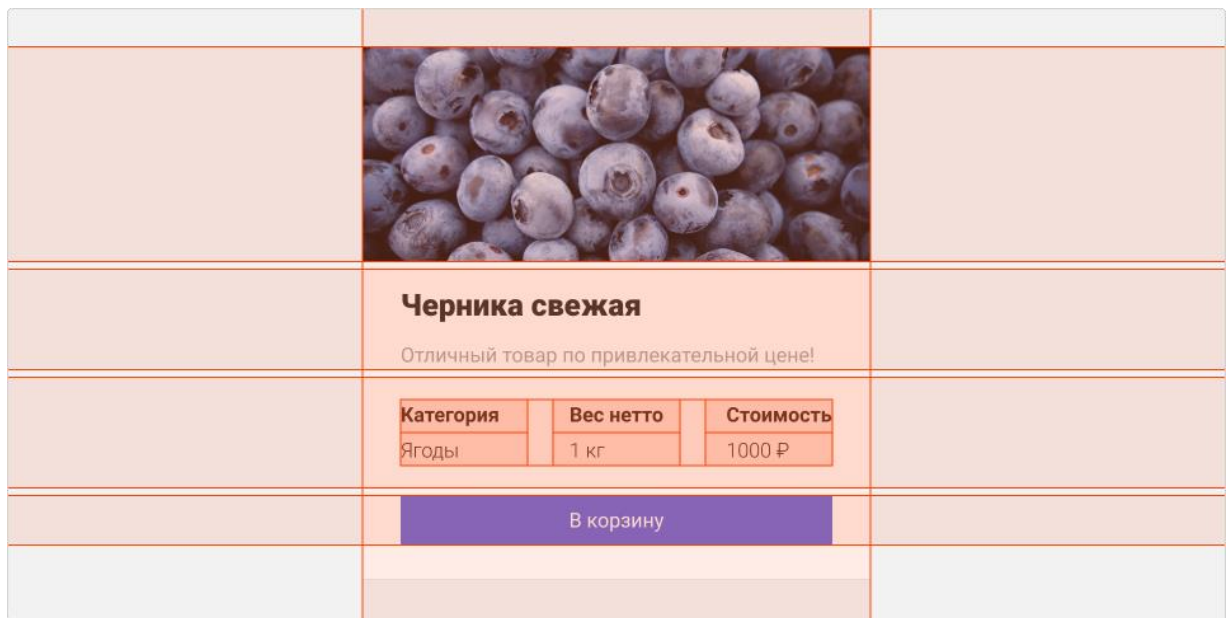
Это понадобится, когда мы будем «переключать» отображение карточки с плиточного в ленточный вид.

Разберём вариант реализации.



Макет одной карточки

Мысленно разделим карточку на отдельные гридовые области:



Макет той же карточки с выведенными направляющими для построения грида

А вот такая будет структура у грида. Общий грид всей карточки будет состоять из четырёх рядов: первый — с фиксированной высотой (так как это изображение), а остальные ряды с автоматической высотой.

Внутри третьего грид-элемента будет вложенный грид со списком определений, похожий на тот, что мы уже разбирали раньше:

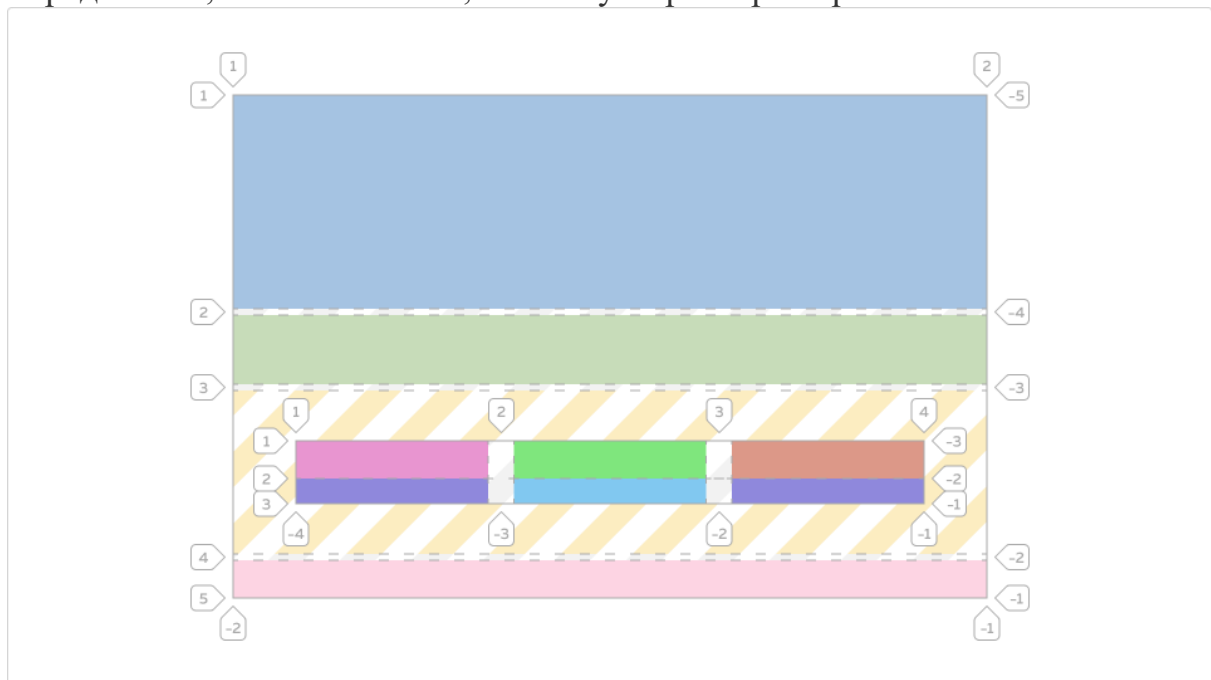


Схема гридов этой карточки, основного и вложенного

Разметка:

```
<article class="card">
```

```
  
```



```
<div class="card-info">

  <h3>Черника свежая</h3>

  <p class="card-info-extra">Отличный товар по привлекательной цене!</p>

</div>


<dl class="card-stats">

  <dt>Категория</dt>

  <dd>Ягоды</dd>

  <dt>Вес нетто</dt>

  <dd>1 кг</dd>

  <dt>Стоимость</dt>

  <dd>1000 ₽</dd>

</dl>


<div class="card-button">

  <a class="button" href="#">В корзину</a>

</div>

</article>
```

Сеточные стили для плиточной раскладки:

```
.card {

  display: grid;

  grid-template-rows: 340px repeat(3, auto);
```

```
    grid-gap: 10px;

    max-width: 350px;
}
```

```
.card-image {

    width: 100%;

    height: 100%;

    object-fit: cover;
}
```

```
.card-stats {

    display: grid;

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

    grid-template-rows: auto 1fr;


    margin-left: 20px;

    margin-right: 20px;
}
```

```
.card-info {

    margin-left: 20px;

    margin-right: 20px;
}
```

```
}

.card-stats dt {

    grid-row: 1 / 2;

}

.card-stats dd {

    margin: 0;

}


.button {

    display: block;

    padding: 10px 20px;

    margin: 20px;

}
```

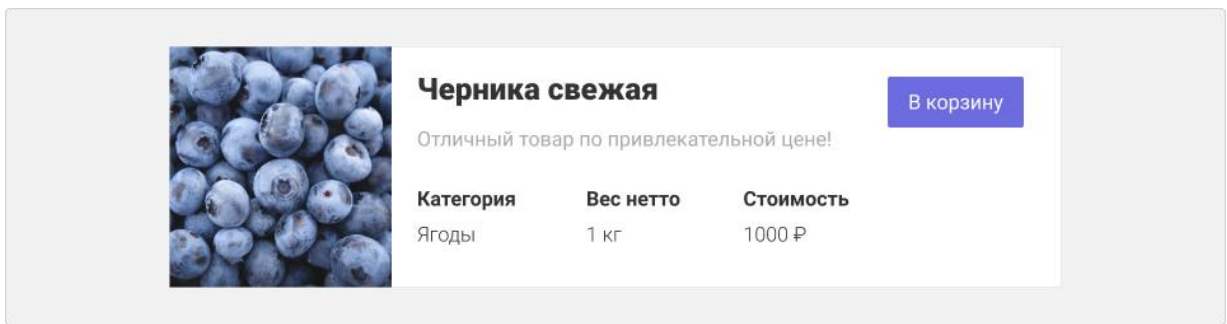
Добавим для элемента `<article class="card">` класс `card-table`.

```
<div><article class="card card-table">



    ...

</article>
```

После добавления класса и изменения раскладки для него карточка должна выглядеть уже так:



Макет той же карточки в табличной раскладке

		<h3>Черника свежая</h3> <p>Отличный товар по привлекательной цене!</p>	В корзину							
		<table><tr><th>Категория</th><th>Вес нетто</th><th>Стоимость</th></tr><tr><td>Ягоды</td><td>1 кг</td><td>1000 ₽</td></tr></table>	Категория	Вес нетто	Стоимость	Ягоды	1 кг	1000 ₽		
Категория	Вес нетто	Стоимость								
Ягоды	1 кг	1000 ₽								

Макет той же карточки с выведенными направляющими для построения грида

Вот такие сеточные стили напишем для табличной раскладки карточки. Теперь в гриде три столбца: первый — с фиксированной шириной, второй — занимает всё оставшееся место, а третий — занимает место по ширине своего контента:

...

```
.card-table {  
  
  grid-template-columns: 350px 1fr auto;  
  
  grid-template-rows: repeat(2, auto);  
  
  max-width: 900px;  
  
}
```

```
.card-table .card-image {  
  
  grid-column: 1 / 2;  
  
  grid-row: 1 / 3;  
  
}
```

```
width: 350px;

height: 250px;

}


.card-table .card-info {

    grid-column: 2 / 3;

    grid-row: 1 / 2;

}


.card-table .card-stats {

    grid-column: 2 / 3;

    grid-row: 2 / 3;

}


.card-table .card-button {

    grid-column: 3 / 4;

    grid-row: 1 / 3;

}
```

Вот такая получилась структура грида:

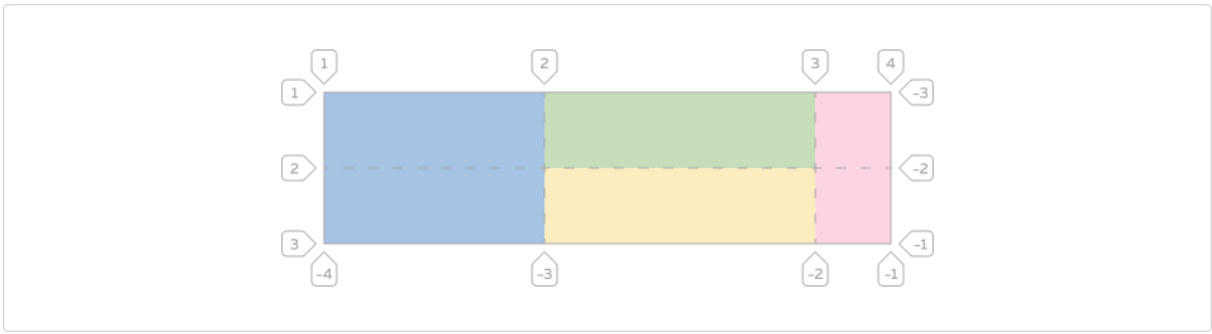
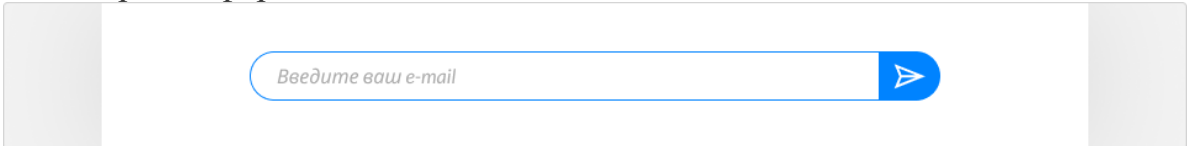


Схема гридов этой карточки в табличной раскладке

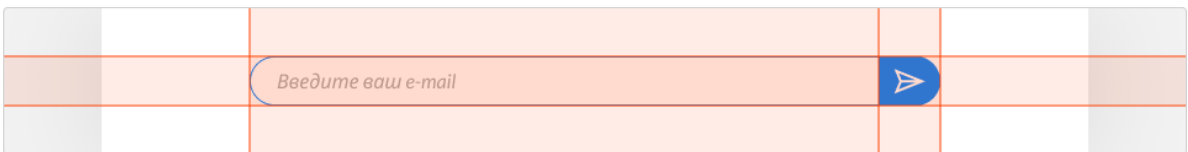
Без внесения в разметку изменений (кроме добавления дополнительного класса самой карточке), только за счёт изменения структуры сетки мы получили новый вид карточки.

Раскладка формы

Применение сеток на гридах для форм начнём с простого примера. Нужно реализовать форму, в которой есть поле для ввода электронной почты и кнопка для отправки формы.



Макет поля для ввода электронного адреса



Макет поля с выведенными направляющими для построения грида



Схема грида для поля

Разметка:

```
<form class="form" action="#">

  <input type="email" placeholder="Ваш email, например, abc@gmail.com">

  <button type="submit">

    ...

  </button>
```

```
</form>
```


Стили:

```
.form {  
  
    display: grid;  
  
    grid-template-columns: 1fr auto;  
  
  
    width: 500px;  
  
    box-sizing: border-box;  
  
}  
  
.form input {  
  
    /* стили поля ввода */  
  
}  
  
.form button {  
  
    /* стили кнопки */  
  
}
```

В примере используется простая сетка, которую можно легко реализовать и другими способами. Но в нашей реализации поле для ввода и кнопка являются грид-элементами, и поэтому высота поля для ввода подстраивается под высоту кнопки, как бы она ни увеличивалась.

Второй пример сетки для формы. На этот раз это форма-анкета для посетителя конференции. Макет более объёмной формы ниже:

Анкета посетителя конференции



Тут будет фото

Имя и фамилия

Место работы

Город

Twitter

Instagram

О себе

Ваша должность

☐ Дизайнер
☐ Разработчик
☐ Менеджер
☐ Другая

Макет формы

Анкета посетителя конференции							
		Имя и фамилия					
		Место работы		Город			
		Twitter		Instagram			
		О себе					
		Ваша должность					
		<input type="checkbox"/> Дизайнер	<input type="checkbox"/> Разработчик	<input type="checkbox"/> Менеджер	<input type="checkbox"/> Другая		

Макет формы с выведенными направляющими для построения грида

Область для заполнения данных можно разделить на пять колонок и шесть строк. Первую колонку занимает область с фотографией участника. Эта колонка занимает две доли свободного пространства (2fr) в отличие от остальных, которые занимают по одной доле (1fr). Поля *Фамилия*, *О себе* и подпись *Ваша должность* занимают по четыре колонки. Поля *Instagram* и *Twitter* — две колонки. Переключатели в поле *Ваша должность* — по одной колонке. Схематично макет формы можно представить в виде:

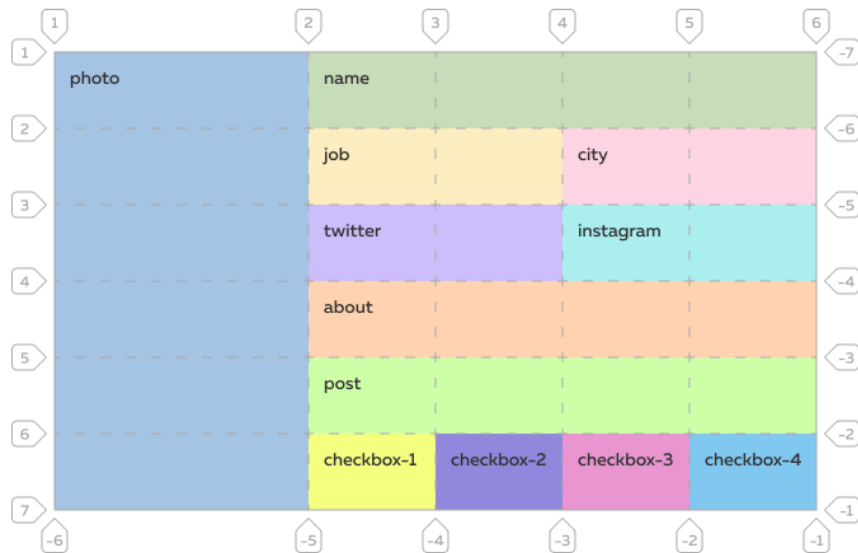


Схема грида для формы

Разметка формы:

```
<section class="profile">

  <h2>Анкета посетителя конференции</h2>

  <form class="profile-form" action="#">

    <div class="photo">

      Тут будет фото

    </div>

    <div class="name">

      <label for="name">Имя и фамилия</label>

      <input type="text" class="input" name="name" id="name">

    </div>
```

```
<div class="job">
```

```
  <label for="job">Место работы</label>
```

```
  <input type="text" class="input" name="job" id="job">
```

```
</div>
```

```
<div class="city">
```

```
  <label for="city">Город</label>
```

```
  <input type="text" class="input" name="city" id="city">
```

```
</div>
```

```
<div class="twitter">
```

```
  <label for="twitter">Twitter</label>
```

```
  <input type="text" class="input" name="twitter" id="twitter">
```

```
</div>
```

```
<div class="instagram">
```

```
  <label for="instagram">Instagram</label>
```

```
  <input type="text" class="input" name="instagram" id="instagram">
```

```
</div>
```

```
<div class="about">
```

```
  <label for="about">0 себе</label>
```

```
<textarea rows="4" class="input" name="about" id="about"></textarea>

</div>


<div class="post">

  <label for="post">Ваша должность</label>

</div>


<div class="checkbox-1">

  <label for="design">

    <input id="design" name="post" type="checkbox" value="1" checked="checked">

    Дизайнер

  </label>

</div>


<div class="checkbox-2">

  <label for="front-end">

    <input id="front-end" name="post" type="checkbox" value="2">

    Разработчик

  </label>

</div>


<div class="checkbox-3">
```

```

        <label for="manager">

            <input id="manager" name="post" type="checkbox" value="3">

            Менеджер

        </label>

    </div>

    <div class="checkbox-4">

        <label for="other">

            <input id="other" name="post" type="checkbox" value="4">

            Другая

        </label>

    </div>

</form>

</section>

```

Сеточные стили для формы:

```

.profile {

    width: 950px;

}

.profile-form {

    display: grid;

    grid-template-columns: 2fr repeat(4, 1fr);

```

```
grid-template-rows: repeat(6, auto);

grid-template-areas:

    "photo name name name name"

    "photo job job city city"

    "photo twitter twitter instagram instagram"

    "photo about about about about"

    "photo post post post post"

    "photo checkbox-1 checkbox-2 checkbox-3 checkbox-4";

grid-gap: 15px;


padding: 15px;

}


.photo {

    grid-area: photo;


    display: flex;

    justify-content: center;

    align-items: center;

}


.name {
```

```
    grid-area: name;  
  
}
```

```
.job {  
  
    grid-area: job;  
  
}
```

```
.city {  
  
    grid-area: city;  
  
}
```

```
.twitter {  
  
    grid-area: twitter;  
  
}
```

```
.instagram {  
  
    grid-area: instagram;  
  
}
```

```
.about {  
  
    grid-area: about;  
  
}
```

```
.post {  
  
  grid-area: post;  
  
}
```

```
.checkbox-1 {  
  
  grid-area: checkbox-1;  
  
}
```

```
.checkbox-2 {  
  
  grid-area: checkbox-2;  
  
}
```

```
.checkbox-3 {  
  
  grid-area: checkbox-3;  
  
}
```

```
.checkbox-4 {  
  
  grid-area: checkbox-4;  
  
}
```

```
.input {
```

```
padding: 10px;

width: 100%;

box-sizing: border-box;

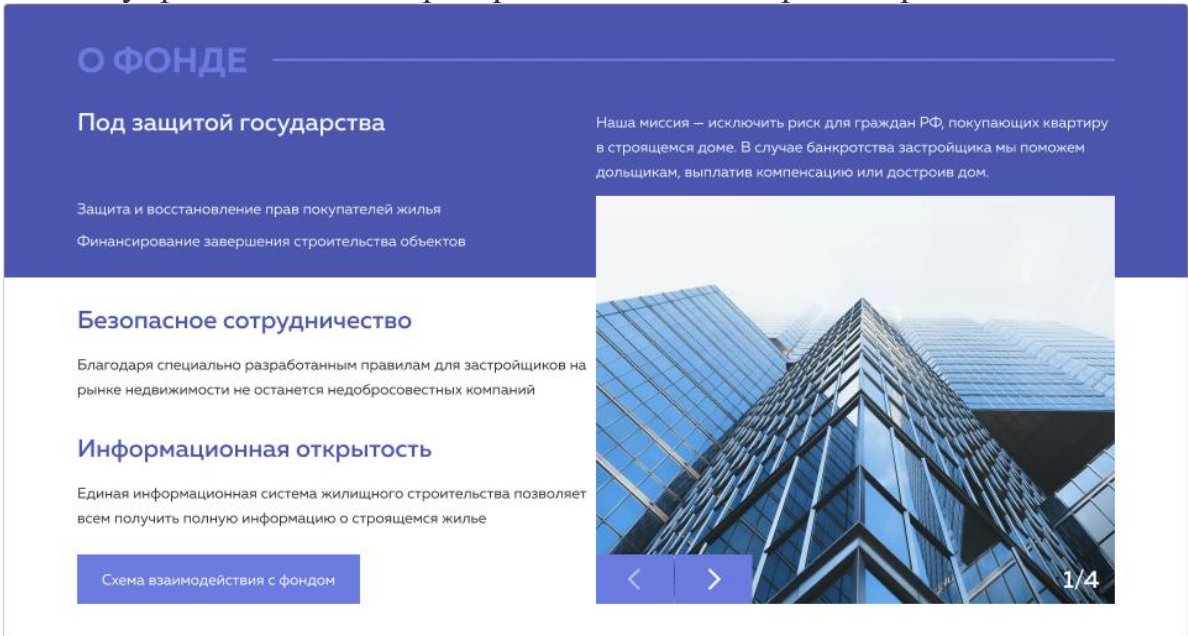
}
```

Раскладка блоков с декоративным наложением друг на друга

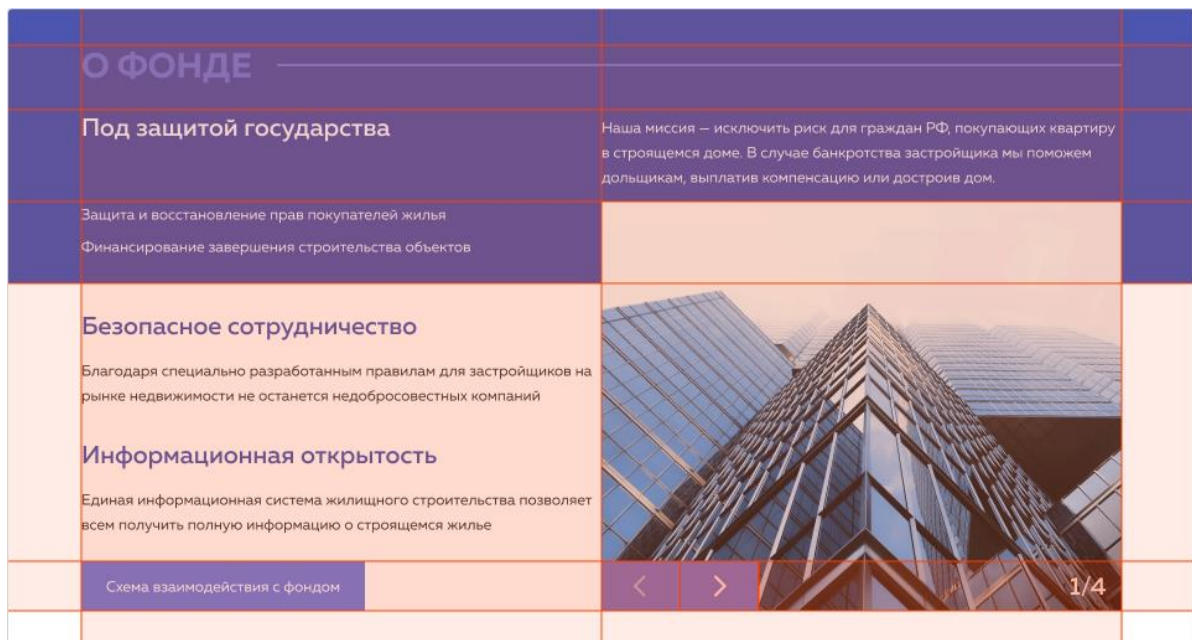
За счёт того, что грид-элементы накладываются друг на друга, можно получать интересные эффекты.

Вот пример типичного раздела страницы лендинга, в котором дизайнер использует многослойность:

- изображение наслаивается на область с текстовым описанием;
- панель управления слайдером расположена поверх изображения.



Макет страницы лендинга



Макет страницы с выведенными направляющими для построения грида

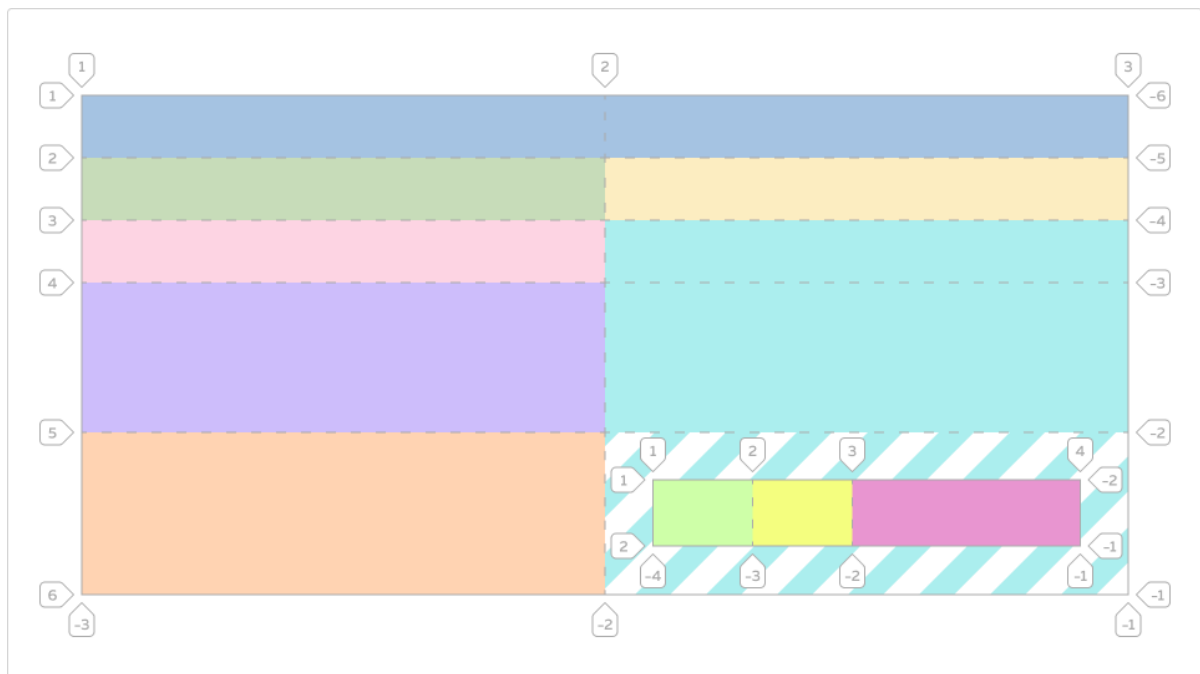


Схема грида для макета страницы

Многослойность легко реализовать с помощью размещения грид-элементов в одной и той же ячейке сетки. Разберём, как это можно сделать. Для начала создадим разметку для раздела.

```
<section class="about container">

  <h2 class="about-tag">О фонде</h2>

  <p class="about-lead">Под защитой государства</p>
```

```
<ul class="about-list">
```

```
<li class="about-item">Защита и восстановление прав покупателей жилья</li>
```

```
<li class="about-item">Финансирование завершения строительства объектов</li>
```

```
</ul>
```

```
<p class="about-desc">Наша миссия – исключить риск для граждан РФ, покупающих
квартиру в строящемся доме. В случае банкротства застройщика мы поможем дольщикам,
выплатив компенсацию или достроив дом.</p>
```

```
<ul class="about-info-block">
```

```
<li class="about-info-block-item">
```

```
<h3 class="about-subtitle">Безопасное сотрудничество</h3>
```

```
<p class="about-text">Благодаря специально разработанным правилам для
застройщиков на рынке недвижимости не останется недобросовестных компаний</p>
```

```
</li>
```

```
<li class="about-info-block-item">
```

```
<h3 class="about-subtitle">Информационная открытость</h3>
```

```
<p class="about-text"><a href="#">Единая информационная система жилищного
строительства</a> позволяет всем получить полную информацию о строящемся жилье</p>
```

```
</li>
```

```
</ul>
```

```
<div class="about-button">

  <a href="#" class="button">Схема взаимодействия с фондом</a>

</div>


<div class="about-img-container">

</div>


<div class="slider-controls">

  <button class="slider-button" type="button">Назад</button>

  <button class="slider-button" type="button">Вперёд</button>

  <span class="slider-page">1 / 4</span>

</div>

</section>
```

Сеточные стили:

```
.container {

  width: 800px;

  margin: 0 auto;

}
```

```
.about {  
  
  display: grid;  
  
  grid-template-columns: 1fr 1fr;  
  
  grid-template-rows: repeat(5, auto);  
  
}
```

```
.about-button {  
  
  grid-area: 5 / 1 / 6 / 2;  
  
  padding: 0 25px;  
  
}
```

```
.about-tag {  
  
  grid-area: 1 / 1 / 2 / 3;  
  
  
  margin: 0;  
  
  padding: 30px 25px;  
  
}
```

```
.about-lead {  
  
  grid-area: 2 / 1 / 3 / 2;
```

```
padding-left: 25px;

}

.about-list {

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

    padding-left: 25px;

}

.about-desc {

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

    padding: 0 25px 20px 0;

}

.about-info-block {

    grid-area: 4 / 1 / 5 / 2;

    padding-left: 25px;

    padding-right: 25px;

    padding-bottom: 30px;

}
```

```
.about-img-container {  
  
    grid-area: 3 / 2 / 6 / 3;  
  
}
```

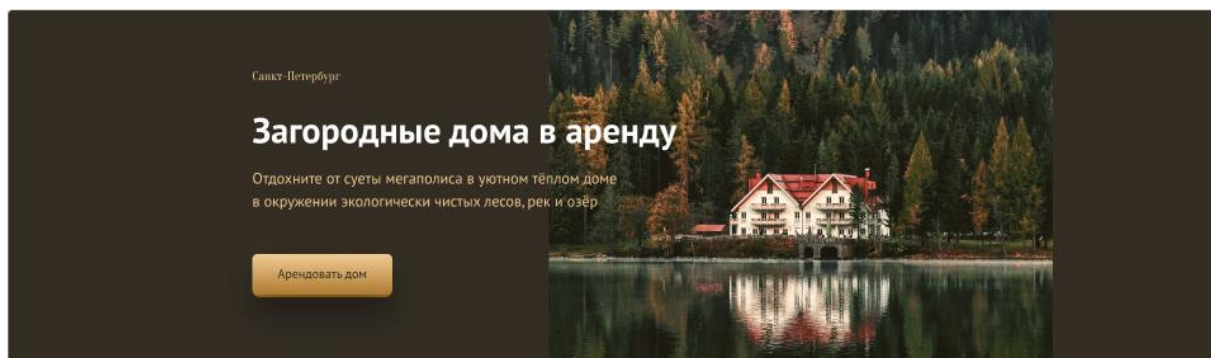
```
.slider-controls {  
  
    /* Грид-элемент является грид-контейнером для растяжения блока с кнопками слайдера  
    по всей ширине доступной области */  
  
    display: grid;  
  
  
    grid-template-columns: auto auto 1fr;  
  
    grid-area: 5 / 2 / 6 / 3;  
  
}
```

```
.about-img {  
  
    /* Оптимальные стили для картинки с кадрированием и автоматической высотой ячейки  
    */  
  
    display: block;  
  
    width: 100%;  
  
    height: 100%;  
  
  
    object-fit: cover;  
  
}
```

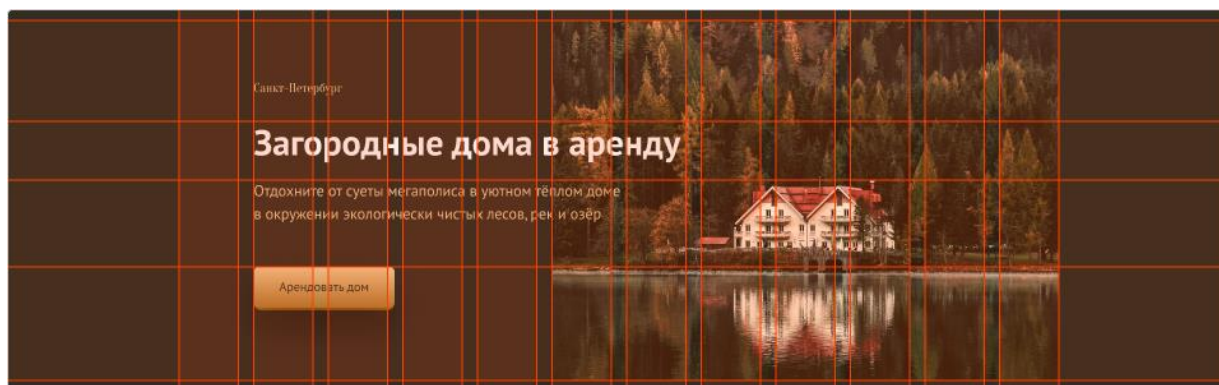
Замечание

Ещё один часто встречающийся кейс в макетах — это наложение текста на изображение. Это может быть блок со слайдером или промо-блок.

Рассмотрим макет с промо-блоком, в котором текст наслаивается на изображение.



Макет промо-блока с наложением текста на картинку



Макет промо-блока с выведенными направляющими для построения грида

Обратите внимание, что сетка имеет двенадцать колонок. Изображение занимает 7 колонок справа, а текст — 6 колонок слева, но располагается начиная со второй колонки. Если два и более элемента размещаются в одной грид-области, то выше будет располагаться тот, который идёт следующим по разметке. Управлять расположением можно с помощью CSS-свойства `z-index`. Подробнее в [углублённой теории о многослойности](#).

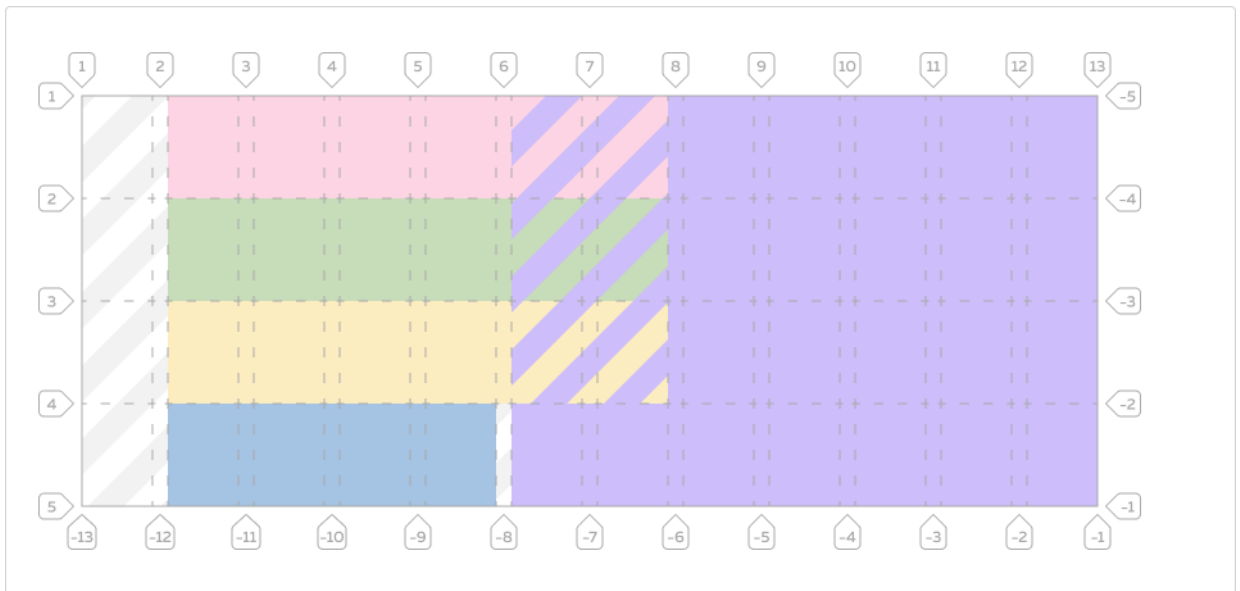


Схема грида для промо-блока

Разметка для блока:

```
<section class="promo">

  <p class="promo-city">Санкт-Петербург</p>

  <h2>Загородные дома в аренду</h2>

  <p class="promo-text">Отдохните от суеты мегаполиса в уютном тёплом доме
  в окружении экологически чистых лесов, рек и озёр</p>

  <a class="promo-button" href="#">Арендовать дом</a>

</section>
```

Стили для блока:

```
.promo {
```



```
display: grid;

grid-template-columns: repeat(12, 1fr);

grid-template-rows: repeat(4, auto);

grid-column-gap: 24px;

}


.promo-city {

    grid-area: 1 / 2 / 2 / 8;

}


.promo h2 {

    grid-area: 2 / 2 / 3 / 8;

    z-index: 1; /* «поднимаем» заголовок над картинкой */

}


.promo-text {

    grid-area: 3 / 2 / 4 / 8;

    z-index: 1; /* «поднимаем» текст над картинкой */

}


.promo-button {

    grid-area: 4 / 2 / 5 / 4;
```

```
}
```

```
.promo-img {
```

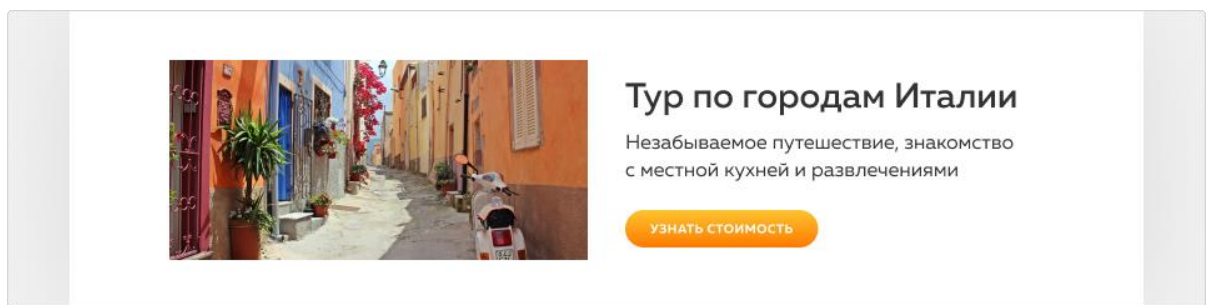
```
  grid-area: 1 / 6 / -1 / -1;
```

```
}
```

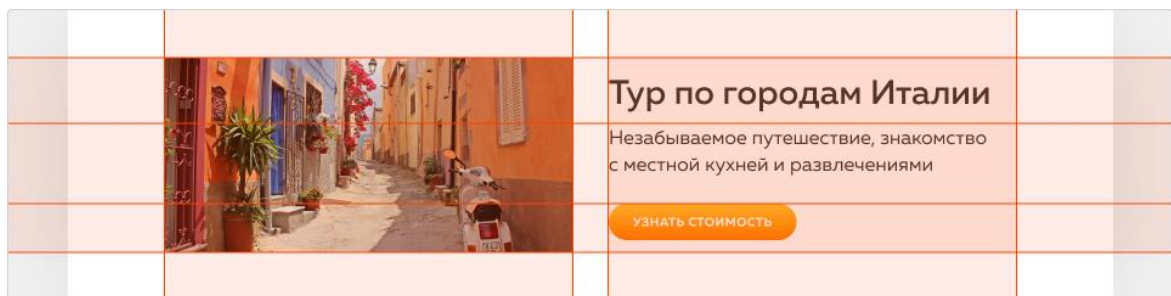
В примере мы воспользовались CSS-свойством `z-index: 1;`, чтобы заголовок и текст «наслоить» на изображение.

Раскладка с изображением и текстом напротив

Без дополнительных обёрток можно построить сетку для элемента такого вида:



Макет карточки



Макет

с выведенными направляющими для построения грида

Получившийся грид состоит из двух столбцов и трёх рядов. Все ряды в первом столбце занимает изображение:

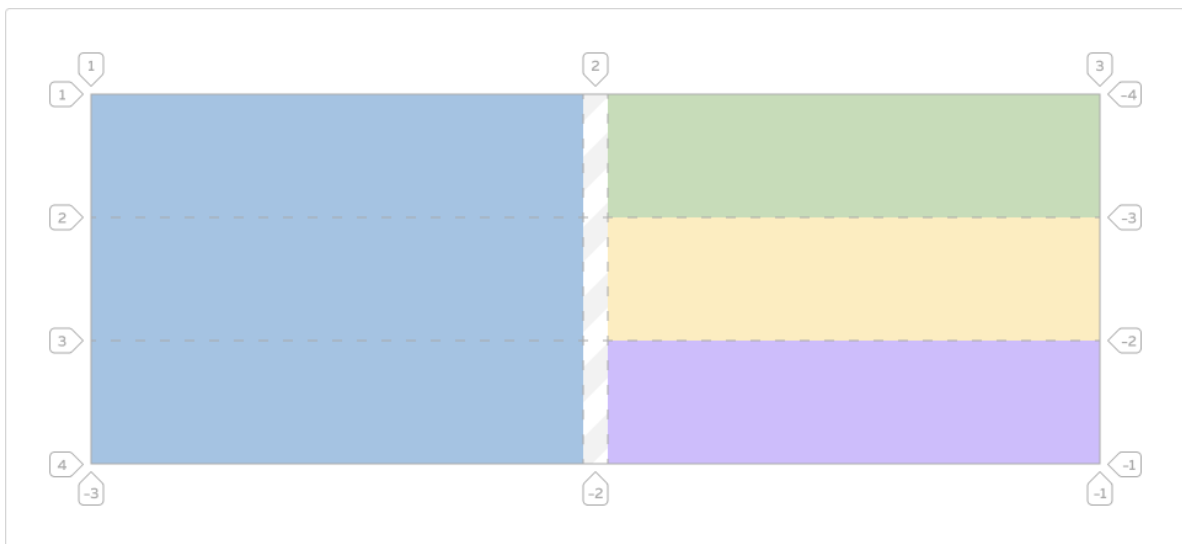


Схема грида для макета

Разметка:

```
<div class="simple-grid">

  <h2 class="simple-grid-name">Тур по городам Италии</h2>

  <p class="simple-grid-subject">Незабываемое путешествие, знакомство с местной
  кухней и развлечениями</p>

  <div class="simple-grid-button">

    <a href="/">Узнать стоимость</a>

  </div>

</div>
```

Стили:

```
.simple-grid {
```

```
display: grid;

grid-template-columns: repeat(2, auto);

grid-template-rows: repeat(3, auto);

grid-template-areas:

    "img name"

    "img subject"

    "img button";

grid-column-gap: 10px;

max-width: 340px;

}
```

```
.simple-grid-img {

    grid-area: img;

}
```

```
.simple-grid-name {

    grid-area: name;

    margin: 0;

}
```

```
.simple-grid-subject {

    grid-area: subject;
```

```
margin: 0;

}

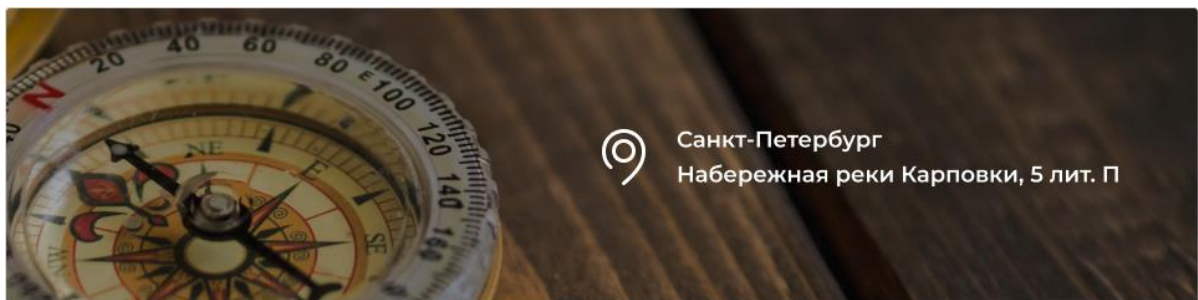
.simple-grid-button {

    grid-area: button;

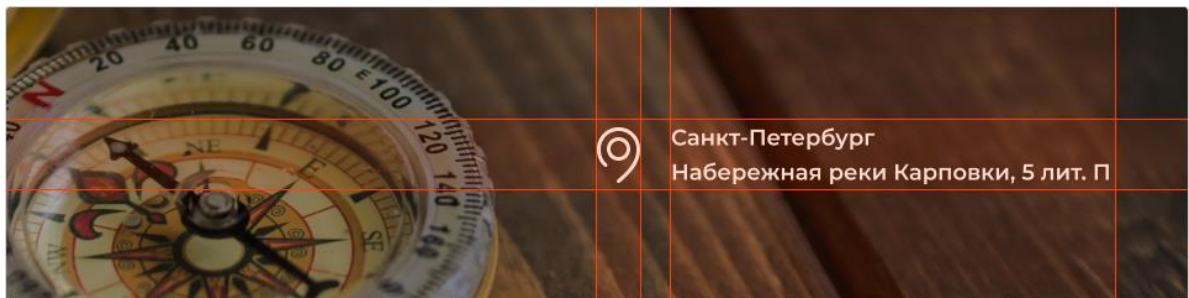
}
```

Раскладка блока с текстом и иконкой

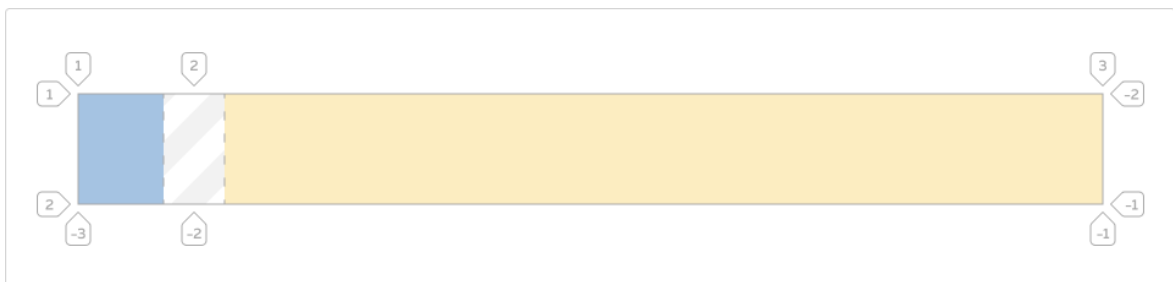
С помощью грида можно легко создавать раскладки с небольшими декоративными элементами: например, с изображением телефона для элемента с телефоном горячей линии, с изображением для пункта меню и так далее. Разберём стилизацию блока с адресом и иконкой напротив.



Макет текста с иконкой



Макет с выведенными направляющими для построения грида



Схема

грида для макета

В разметке элемент выглядит следующим образом:

```
<address class="address">Санкт Петербург, набережная реки Карповки, 5 лит.  
П</address>
```

Добавим декоративный элемент с помощью

псевдоэлемента `::before` и применим `display: grid;` для тега `<address>`:

```
.address {  
  
    display: grid;  
  
    grid-template-columns: auto auto;  
  
    grid-column-gap: 50px;  
  
}  
  
.address::before {  
  
    content: "";  
  
    width: 16px;  
  
    height: 20px;  
  
    background-image: url("img/pin.svg");  
  
    background-repeat: no-repeat;  
  
    background-position: center;  
  
}
```

Шаг 4. Добавляем выравнивание

Вроде бы мы всё сделали правильно: грид создан, элементы в нём выстроены в правильном порядке, но почему-то не всё соответствует макету. Особенно это заметно, если элементы грида разнородные.

Вот, например, достаточно типичная шапка сайта. В ней есть логотип сайта, основное меню, форма поиска и пользовательская навигация.

```
<header class="page-header">
  <nav class="page-nav container">
    <a class="logo" href="#">
      
    </a>
    <ul class="site-nav">
      <li class="site-nav-item">
        <a class="site-nav-link page-link" href="#">Не видеть зла</a>
      </li>
      <li class="site-nav-item">
        <a class="site-nav-link page-link" href="#">Не слышать зла</a>
      </li>
      <li class="site-nav-item">
        <a class="site-nav-link page-link" href="#">Не говорю зло</a>
      </li>
    </ul>
    <form class="search-form" action="#" method="post">
      <input type="search" name="query" placeholder="Поиск по сайту">
      <button type="submit">Найти</button>
    </form>
    <p class="user-nav">
      <a class="page-link" href="#">Вход</a> или <a class="page-link"
href="#">регистрация</a>
    </p>
  </nav>
</header>
```

Если выстроить эти элементы с помощью грида, то они, конечно, встанут в ряд.

```
.page-nav {
  display: grid;
  grid-template-columns: auto 1fr auto auto; /* довольно очевидно, что
всё возможное место должно достаться меню */
  grid-column-gap: 25px;
}

.site-nav {
  padding: 0;
  margin: 0;
  list-style: none;
  display: flex; /* как красиво встать эти элементы гридом — покажем
на сложном уровне */
  flex-wrap: wrap; /* перенос в таких случаях обязателен */
}
```

```

/* промежуток между элементами меню */
.site-nav-item {
  padding-right: 25px;
}

.site-nav-item:last-child {
  padding-right: 0;
}

.page-link {
  /* это просто текстовые параметры, чтобы в глазах не рябило */
  text-decoration: none;
  font-weight: normal;
  font-size: 16px;
  line-height: 20px;
  color: inherit;
}

```

Но — это не будет красиво, грид-элементы из-за своих особенностей будут на разной высоте.



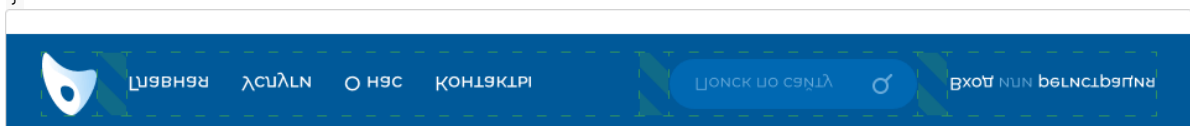
Грид-элементы выстроились в ряд и заняли положенные места, но есть нюанс

И в этот момент нам на помощь приходят выравнивания. Все значения описали в теоретическом материале.

```

.page-nav {
  display: grid;
  grid-template-columns: auto 1fr auto auto;
  grid-column-gap: 25px;
  align-items: center;
}

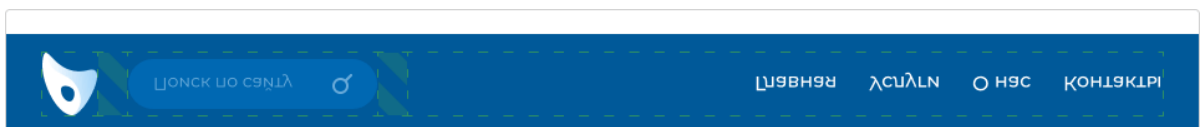
```



Грид-элементы на своих местах и по центру относительно вертикальной оси

Если бы после этого мы захотели один из элементов выровнять как-то иначе, например, прибить к низу контейнера, мы бы использовали свойство `align-self`.

Второй случай, в котором пригодятся выравнивания — если мы хотим оттолкнуть один из элементов. В том же макете поиск мог бы быть прямо рядом с логотипом, пользовательской навигации могло бы вовсе не быть, а меню могло бы прилипнуть к правому краю.



Индивидуальное выравнивание вдоль горизонтальной оси

В таком случае нужно заставить логотип и форму равняться по левому краю, а навигацию — по правому.

Разметка грида


```

<header class="page-header"> <nav class="page-nav container"> <a
class="logo" href="#">  </a> <form class="search-form" action="#"
method="post"> <input type="search" name="query" placeholder="Поиск по
сайту"> <button type="submit">Найти</button> </form> <ul class="site-
nav"> <li class="site-nav-item"> <a class="site-nav-link page-link"
href="#">Главная</a> </li> <li class="site-nav-item"> <a class="site-
nav-link page-link" href="#">Услуги</a> </li> <li class="site-nav-
item"> <a class="site-nav-link page-link" href="#">О нас</a> </li> <li
class="site-nav-item"> <a class="site-nav-link page-link"
href="#">Контакты</a> </li> </ul> </nav> </header>
.page-nav {
  display: grid;
  grid-template-columns: auto auto 1fr;
  grid-column-gap: 25px;
  align-items: center;
  justify-items: start; /* все элементы выстроились в левому краю
контейнера */
}

.site-nav {
  justify-self: end; /* навигация прилипла к правому краю контейнера
*/
}

```

Помимо того, что выравнивание выстраивает разнотипные элементы, за счёт выравнивание можно избавиться от лишних обёрток в сетке. Разберём типовой случай.

У нас есть промо-блок с ссылкой «Записаться на обучение».

```

<section class="promo">
  <a href="#">
    
  </a>
  <h2>Обучаем управлению парусным судном</h2>
  <p class="promo-text">с нуля до помощника капитана за 1 месяц</p>
  <a class="button" href="#">Записаться на обучение</a>
  
</section>

```

Постоим сетку для раздела и распределим элементы по ней:

```

.promo {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(4, auto);
  grid-gap: 10px 30px;
  width: 950px;
}

.promo h2 {
  grid-area: 2 / 1 / 3 / 3;
}

.promo-text {
  grid-area: 3 / 1 / 4 / 3;
}

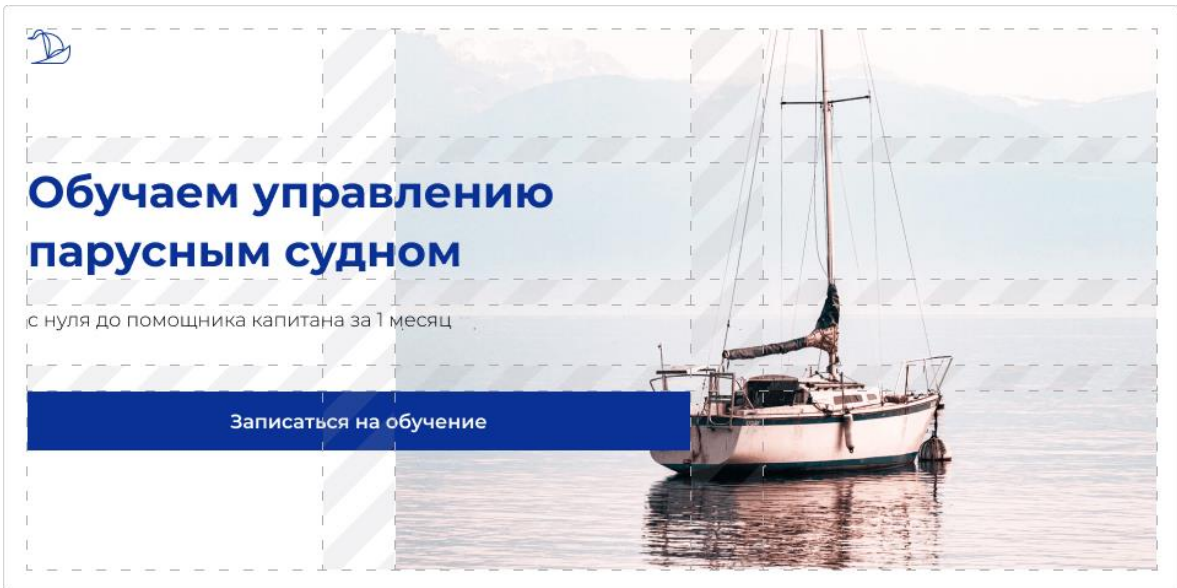
.promo .button {
  grid-area: 4 / 1 / 5 / 3;
}

```

```
padding: 16px 32px;
color: #ffffff;
background-color: #06a9b3;
text-decoration: none;
text-align: center;
border: none;
}

.promo img {
  grid-area: 1 / 3 / 5 / 4;
}
```

По умолчанию ссылка «Записаться на обучение» растягивается на всю ширину грид-ячеек.



По умолчанию элементы растягиваются на всю ширину грид-ячеек

Но по макету элемент занимает меньшую ширину. Есть два варианта решения этой задачи. Первый, добавить обёртку над ссылкой, но при этом мы теряем преимущества использования гридов (создание сетки без лишних обёрток). Второй, добавить выравнивание. После этого размеры элемента соответствуют макету и мы не используем дополнительные элементы.

```
.promo {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(4, auto);
  grid-gap: 10px 30px;
  width: 950px;
}

.promo h2 {
  grid-area: 2 / 1 / 3 / 3;
}

.promo-text {
  grid-area: 3 / 1 / 4 / 3;
}

.promo .button {
  grid-area: 4 / 1 / 5 / 3;
  align-self: start;
  justify-self: start;
  padding: 16px 32px;
}
```

```

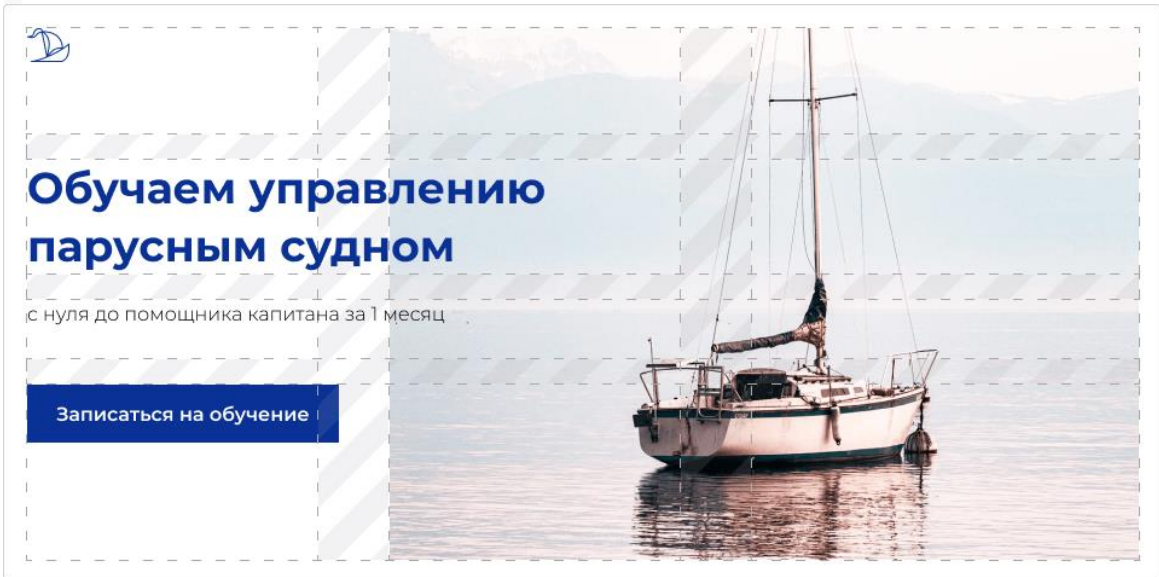
color: #ffffff;
background-color: #06a9b3;
text-decoration: none;
text-align: center;
border: none;
}

```

```

.promo img {
  grid-area: 1 / 3 / 5 / 4;
}

```



Изменяем выравнивание по умолчанию

Баг с выравниванием изображения по умолчанию

Возможно, вы замечали, если поместить изображение в контейнер, в котором больше ничего нет, то между нижним краем контейнера и изображением будет небольшое расстояние. Так себя ведёт стандартное выравнивание по вертикали `vertical-align: baseline` для тега ``.

```

<section class="card">
  <figure>
    
  </figure>
  <h2>Домашние растения в ассортименте</h2>
  <p class="text">Широкий выбор растений и цветов со скидкой от 90%.
  Закажите сейчас, ведь завтра будет уже поздно.</p>
  <a class="button" href="#">Заказать</a>
</section>
.card {
  display: grid;
  grid-template-columns: auto 1fr;
  grid-template-rows: repeat(3, auto);
  grid-column-gap: 30px;
  background-color: #eeeeee;
}

```

```

.card figure {
  grid-area: 1 / 1 / 4 / 2;
  margin: 0;
}

```

```
}
```

```
...
```

Шаг 5*. Используем автоматическое размещение и заполнение

Автоматическое размещение

На практике автоматическое размещение или свойство `grid-auto-flow` может быть полезным, когда страница заполняется на основе данных, которые приходят с сервера, причём эти данные при отображении разного размера и мы не управляем их порядком.

К примеру, у нас есть каталог товаров. Часть товаров нужно выделить, допустим, показать в карточке большего размера. Разметка каталога выглядит следующим образом:

```
<ul class="catalog-list">

  <li class="catalog-item">

    <section class="card">

      <h2>Рюкзак BeastCraft</h2>

      <p>1500₽</p>

    </section>

  </li>

  <li class="catalog-item">

    <section class="card">

      <h2>Рюкзак походный 35л</h2>
```

```
<p>2000₽</p>



</section>

</li>

<li class="catalog-item catalog-item-big">

  <section class="card">

    <h2>Сумка спортивная</h2>

    <p>1700₽</p>

  </section>

</li>

<li class="catalog-item">

  <section class="card">

    <h2>Рюкзак Fjallraven Kanken</h2>

    <p>4200₽</p>

  </section>

</li>

<li class="catalog-item">

  <section class="card">

    <h2>Рюкзак Langly</h2>

    <p>3900₽</p>
```

```
        

    </section>

</li>

<li class="catalog-item">

    <section class="card">

        <h2>Рюкзак Haupes</h2>

        <p>3300Р</p>

    </section>

</li>

<li class="catalog-item">

    <section class="card">

        <h2>Сумка кожаная</h2>

        <p>5000Р</p>

    </section>

</li>

<li class="catalog-item">

    <section class="card">

        <h2>Рюкзак Vinta</h2>

        <p>2900Р</p>

        
```

```

    </section>

</li>

<li class="catalog-item">

    <section class="card">

        <h2>Рюкзак детский</h2>

        <p>800₽</p>

    </section>

</li>

</ul>

```

Карточку большего размера выделим классом `catalog-item-big`.

С помощью автоматического заполнения мы создадим сетку, в которой будут размещаться карточки разного размера, не будет пустых ячеек и при этом в стилях добавиться лишь одно CSS-свойство. Браузер всё сделает за нас.

```

.catalog-list {

    list-style: none;

    padding: 0;

    margin: 0;

    display: grid;

    grid-gap: 66px;

    grid-template-columns: repeat(4, 243px);

    grid-auto-flow: dense;

```

```
}
```

```
.catalog-item-big {
```

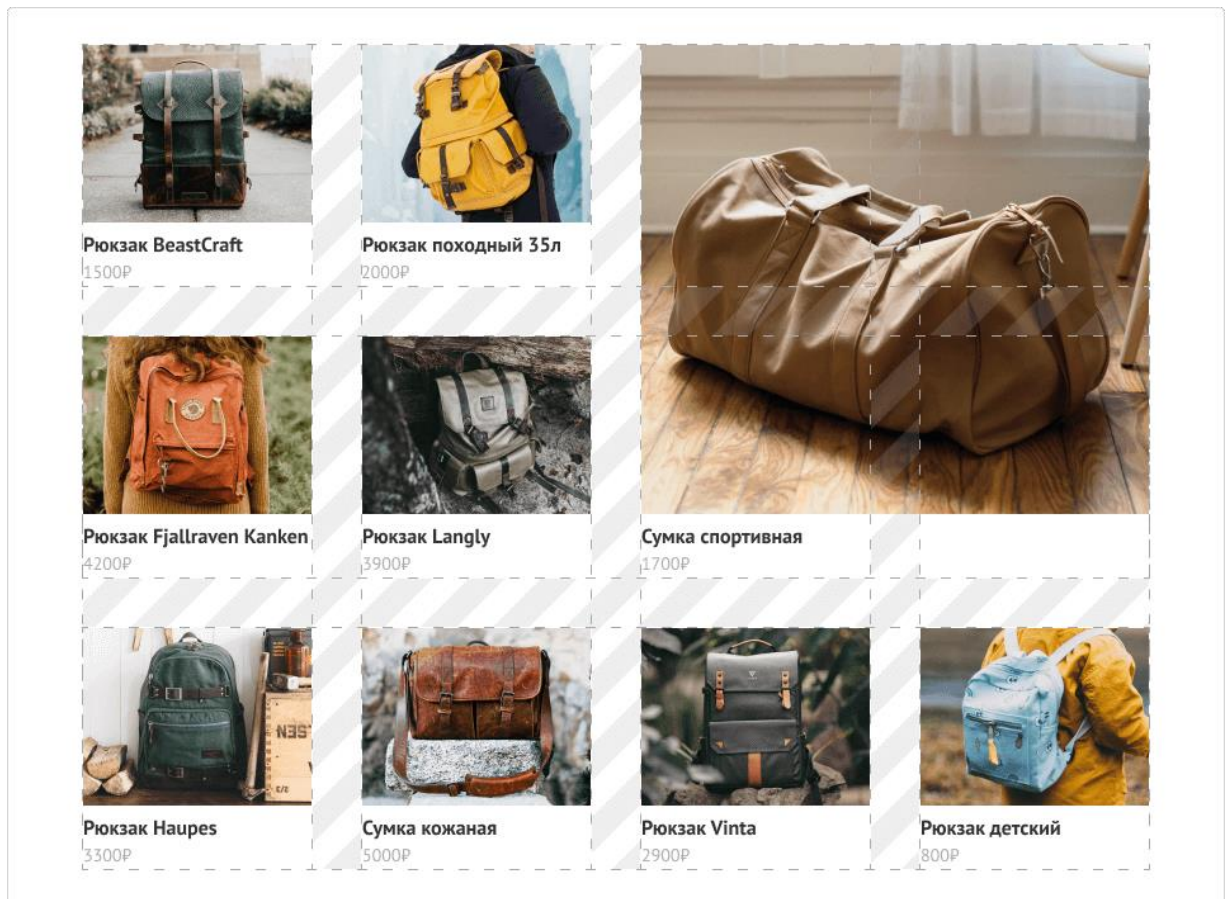
```
  grid-column: span 2;
```

```
  grid-row: span 2;
```

```
}
```

```
...
```

Результат выглядит так:



grid-auto-flow: dense

Автоматическое заполнение

Автоматическое заполнение чаще всего используется для создания адаптивных сеток. Но значение `auto-fit` можно использовать, если у нас

заранее не известно количество элементов и нужно их растянуть по всей ширине внешнего контейнера.

Например, список хештегов для изображения:



auto-fit для списка хештегов

Разметка блока:

```
<section class="post">

  <ul class="hashtag-list">

    <li class="hashtag">

      <a href="#">#show</a>

    </li>

    <li class="hashtag">

      <a href="#">#cloudyday</a>

    </li>

    <li class="hashtag">

      <a href="#">#wintertime</a>

    </li>

    <li class="hashtag">

      <a href="#">#showball</a>

    </li>
```

```
</ul>

...

</section>
```

Стили:

```
.hashtag-list {

  list-style: none;

  padding: 0;

  margin: 0;

  display: grid;

  grid-gap: 10px;

  grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));

}

.hashtag {

  padding: 10px 20px;

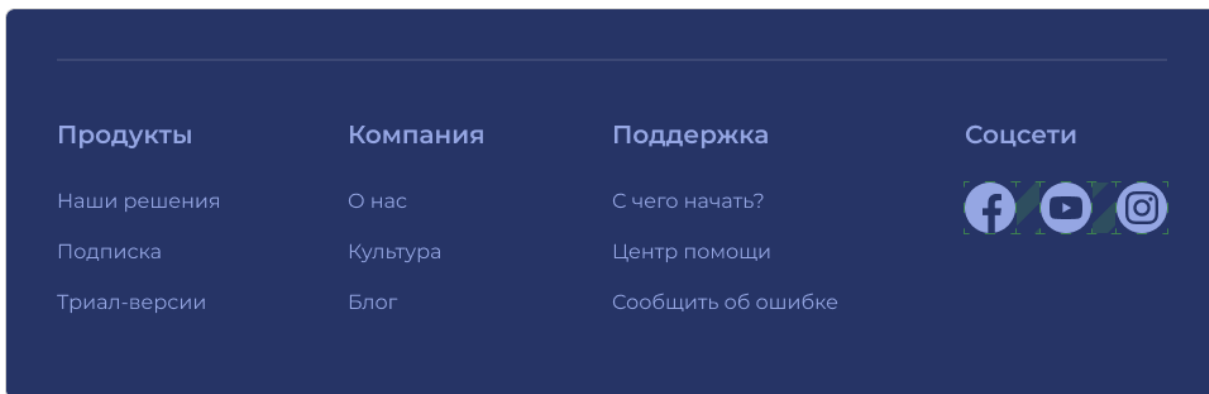
  text-align: center;

  background-color: #eeeeff;

}

...
```

Ещё один частый кейс, в котором можно применить значение `auto-fit` — это сетка для списка ссылок на страницы в социальных сетях.



`auto-fit` для списка ссылок на страницы в социальных сетях

Как правило, ссылки на страницы с социальных сетей отображаются в виде иконок одного размера. У *CSS Grid Layout* в этом случае есть преимущество перед *CSS Flex Layout*, не нужно сбрасывать отступ у последнего элемента (`grid-gap`). А за счёт `auto-fit` можно предусмотреть переполнение элементов, к примеру, организация зарегистрировалась в новой соцсети и нужно добавить соответствующую ссылку. Единственное ограничение, элемент со списком должен быть ограничен по ширине.

```
<ul class="social-links">

  <li class="social-links-item social-links-item-facebook">

    <a href="#">

    </a>

  </li>

  <li class="social-links-item social-links-item-youtube">

    <a href="#">

    </a>

  </li>

  <li class="social-links-item social-links-item-instagram">
```

```

        <a href="#">

        </a>

    </li>

</ul>

.social-links {

    display: grid;

    grid-template-columns: repeat(auto-fit, 32px);

    column-gap: 20px;

    margin: 0;

    padding: 0;

    list-style: none;

    justify-self: end;

    max-width: 500px;

}

...

```

Добавим новые ссылки на *Twitter* и *ВКонтакте*.

```

<ul class="social-links">

    <li class="social-links-item social-links-item-facebook">

        <a href="#">

```

```
</a>

</li>

<li class="social-links-item social-links-item-youtube">

  <a href="#">

  </a>

</li>

<li class="social-links-item social-links-item-instagram">

  <a href="#">

  </a>

</li>

<li class="social-links-item social-links-item-twitter">

  <a href="#">

  </a>

</li>

<li class="social-links-item social-links-item-vk">

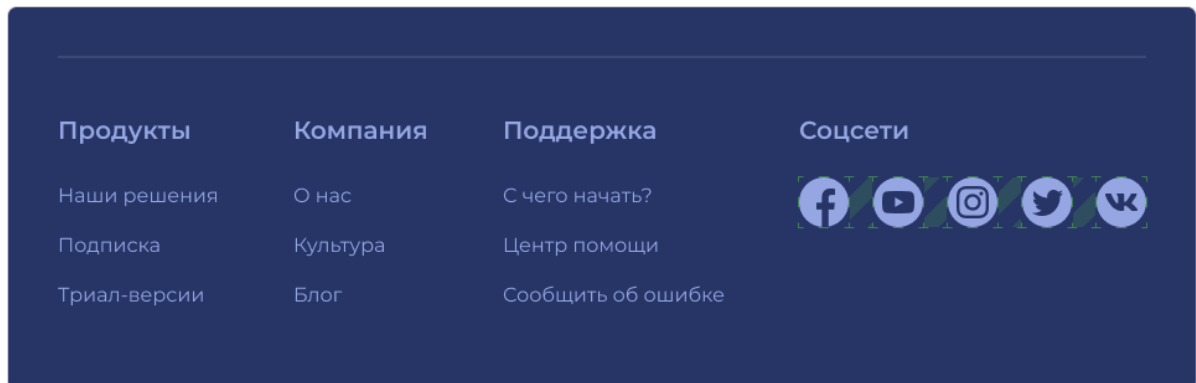
  <a href="#">

  </a>

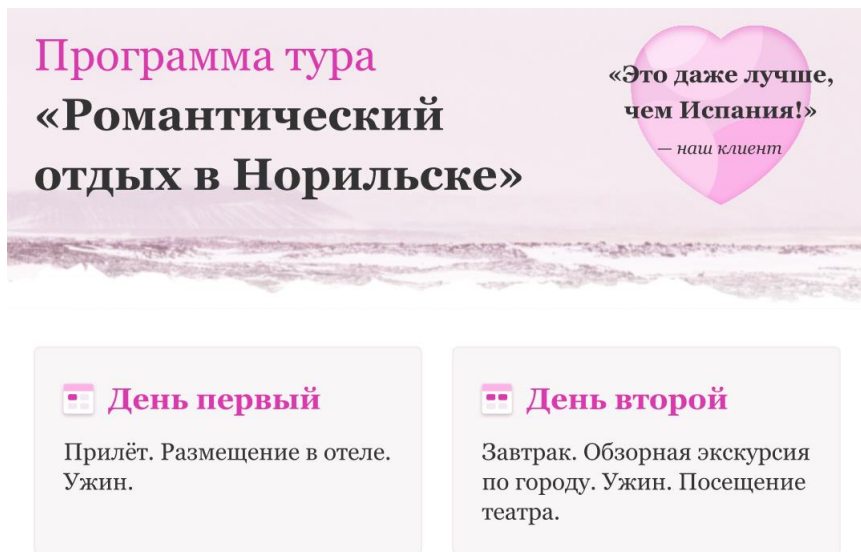
</li>
```


Элементы заняли своё место в ряду ссылок, при этом наши стили не изменились.



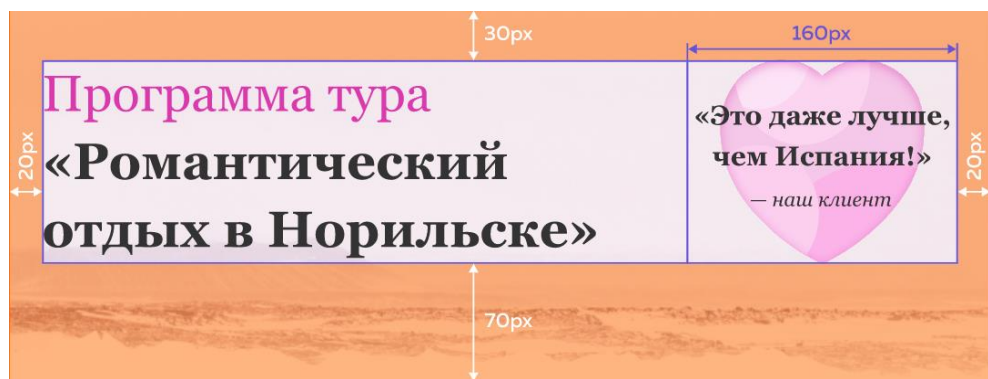
новые элементы в списке ссылок на страницы в социальных сетях

Задание 2. Самостоятельно сверстайте сетку страницы «Программа тура»

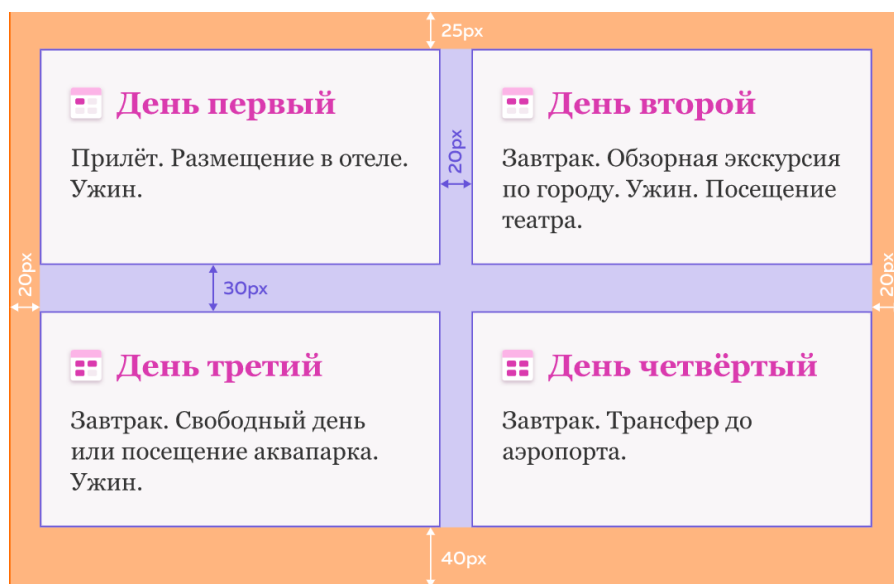


Разметка и декоративные стили готовы, осталось написать сеточные стили. Советуем сразу обнулить внешние отступы у `<body>`, а для создания колонок использовать гриды.

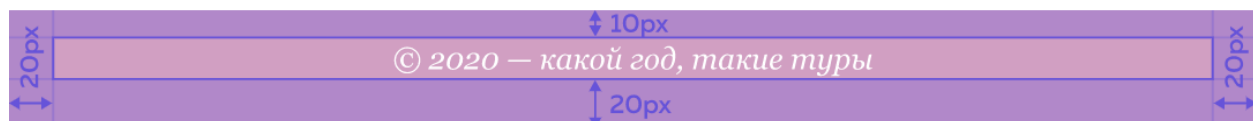
В шапке должно быть две колонки: правая имеет фиксированный размер, а левая занимает всё оставшееся пространство. Не забудьте про внутренние отступы.



Список дней также нужно разделить на две колонки, на этот раз одинаковой ширины. Колонки должны занимать всё доступное пространство. Не забудьте убрать у списка внешние отступы по умолчанию и переопределить внутренние отступы. Обратите особое внимание на отступы между элементами, они разные у рядов и колонок.

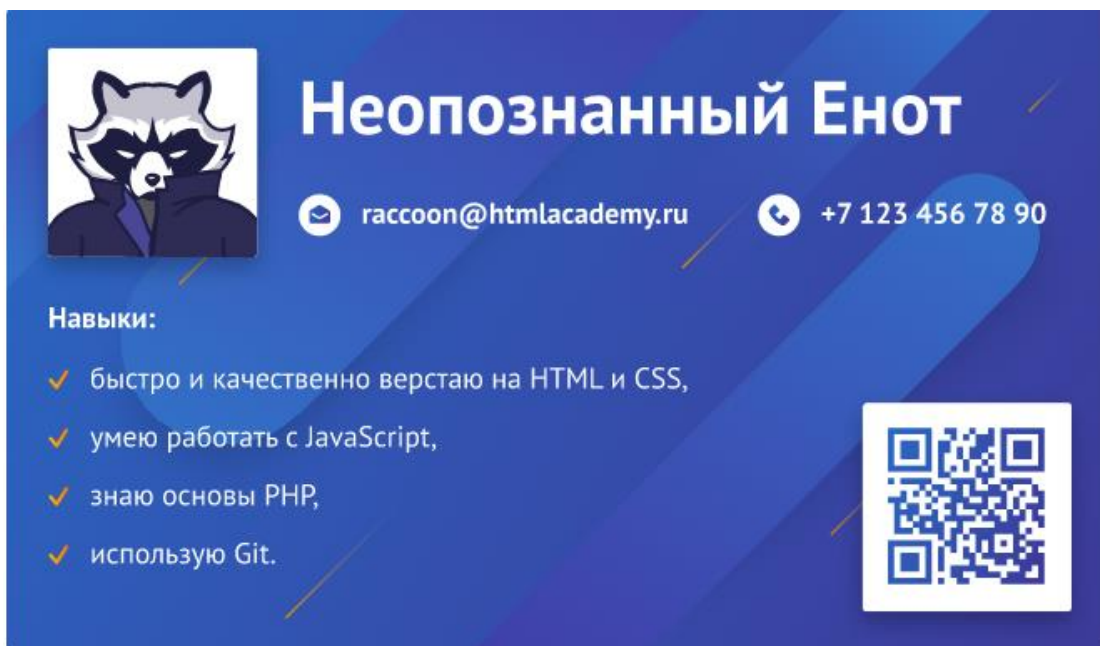


Подвал страницы довольно простой, но у него отличается внутренний отступ сверху.

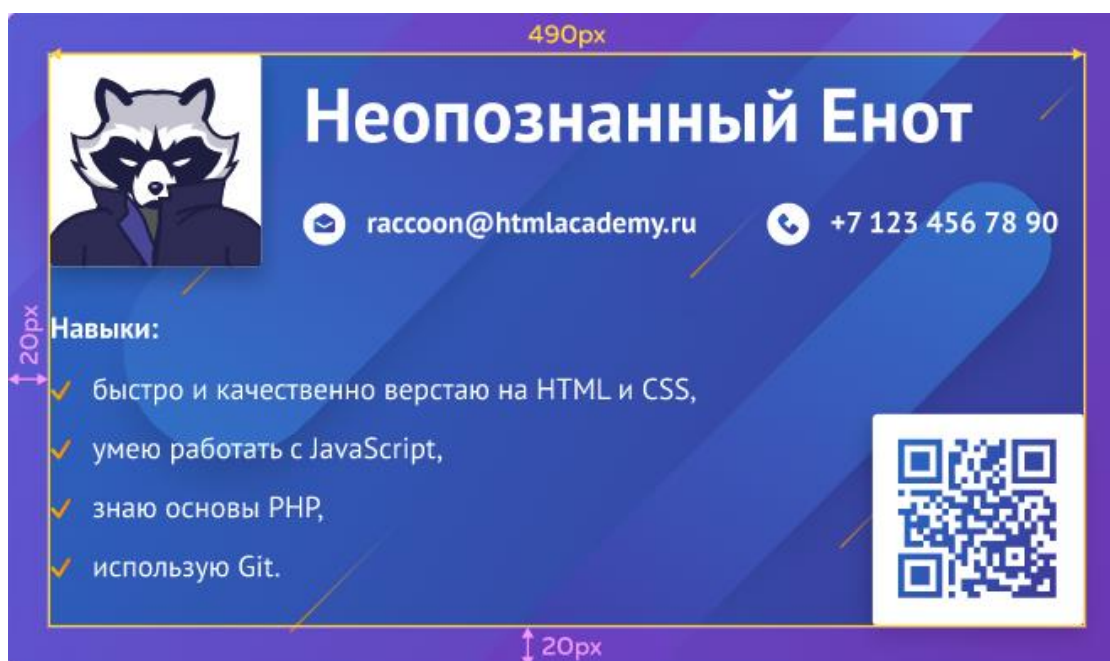


Задание 3. визитка Неопознанного Енота

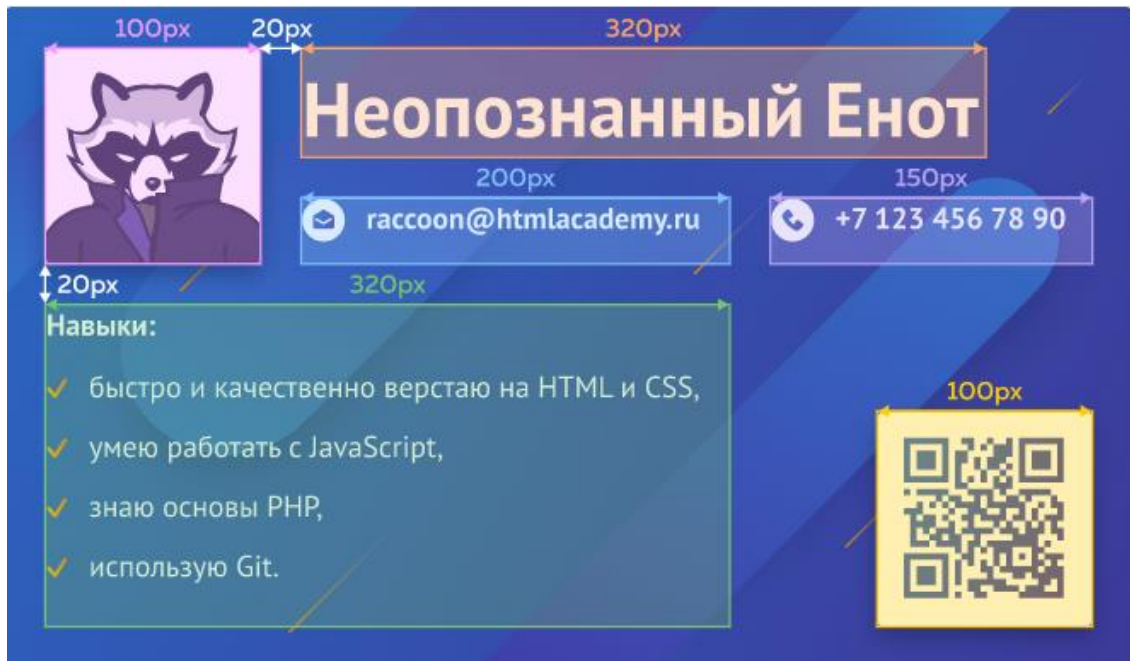
В этом испытании вам предстоит сверстать визитную карточку. Все декоративные стили уже готовы, осталось только описать сетку карточки.



Визитка должна иметь фиксированную ширину и внутренние отступы со всех сторон.



Элементы в карточке расположены асимметрично. Подумайте, сколько колонок и какой ширины понадобится, чтобы сверстать эту карточку. Не забудьте про отступы между рядами и колонками.



Обратите внимание, порядок элементов в разметке не соответствует порядку элементов на макете. Используйте грид-области и выравнивание, чтобы правильно разместить элементы.

Дополнительные материалы

<https://studfile.net/preview/4218415/>

<https://studfile.net/preview/9202055/page:8/>