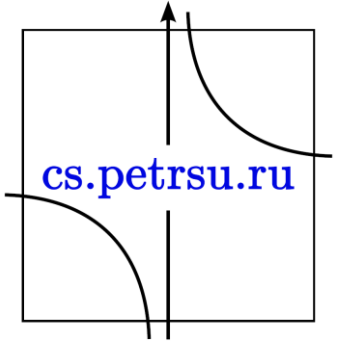
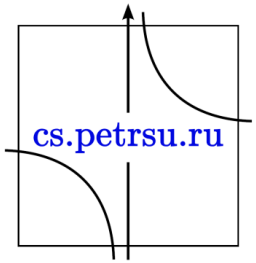


HTML5
Display: table,grid,flex

Лекция №3

HTML5

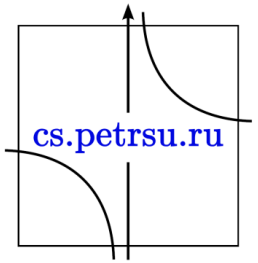
- Организация WHATWG (Web Hypertext Application Technology Working Group, Рабочая группа по разработке гипертекстовых приложений Интернета).
  - разработчики браузеров Safari, Firefox и Opera
- Идеи W3C, современные потребности пользователей и мнения веб-разработчиков воплотились в новом языке разметки названном HTML5.
- Опубликован в 2014 году



HTML5

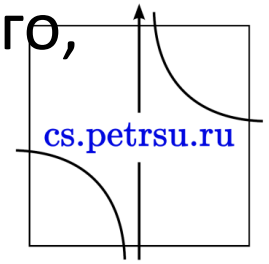
— это не продолжатель языка разметки гипертекста, а новая открытая платформа, предназначенная для создания веб-приложений использующих аудио, видео, графику, анимацию и многое другое.

- Цель разработки HTML5
  - улучшение уровня поддержки мультимедиа-технологий
  - сохранение обратной совместимости
  - Улучшение удобочитаемости кода для человека и простоты анализа для парсеров.



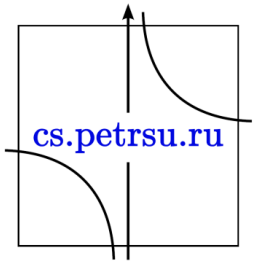
HTML5

- В HTML5 реализовано множество новых синтаксических особенностей. Например, элементы <video>, <audio> и <canvas>, а также возможность использования SVG и математических формул.
- Новшества разработаны для упрощения создания и управления графическими и мультимедийными объектами в сети без необходимости использования сторонних API и плагинов.
- Другие новые элементы, такие как <section>, <article>, <header> и <nav>, разработаны для того, чтобы обогащать семантическое содержимое документа.



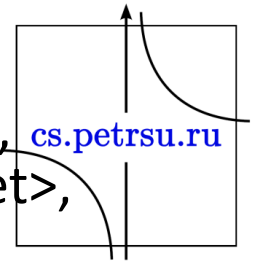
HTML5

- Некоторые элементы, например <a>, <menu> и <cite>, были изменены, переопределены или стандартизированы.
- API и DOM стали основными частями спецификации HTML5.
- HTML5 также определяет некоторые особенности обработки ошибок вёрстки, поэтому синтаксические ошибки должны рассматриваться одинаково всеми совместимыми браузерами.



Отличия HTML5 от HTML4.01 и XHTML1.0

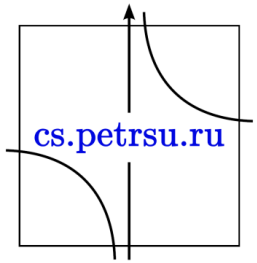
- Изменён синтаксис
- Поддержка геолокации
- Встраивание SVG и MathML в text/html
- Новые элементы: <article>, <aside>, <audio>, <canvas>, <command>, <datalist>, <details>, <embed>, <figcaption>, <figure>, <footer>, <header>, <hgroup>, <keygen>, <main>, <mark>, <meter>, <nav>, <output>, <progress>, <rp>, <rt>, <ruby> (англ.), <section>, <source>, <summary>, <time>, <video>, <wbr>
- Новые компоненты ввода: date/time, email, url, search, number, range, tel, color
- Новые атрибуты: charset (в <meta>), async (в script)
- Глобальные атрибуты, которые могут быть применены ко всем элементам: id, tabindex, hidden, data-* (пользовательские атрибуты данных)
- Элементы, которые будут исключены: <acronym>, <applet>, <basefont>, <big>, , <dir>, <font>, <frame>, <frameset>, <isindex>, <noframes>, <strike>, <tt>



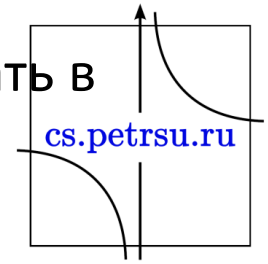
Изменения касательно структуры кода

- <!DOCTYPE html>

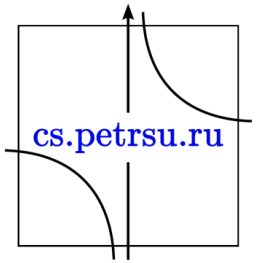
- <meta charset="utf-8">



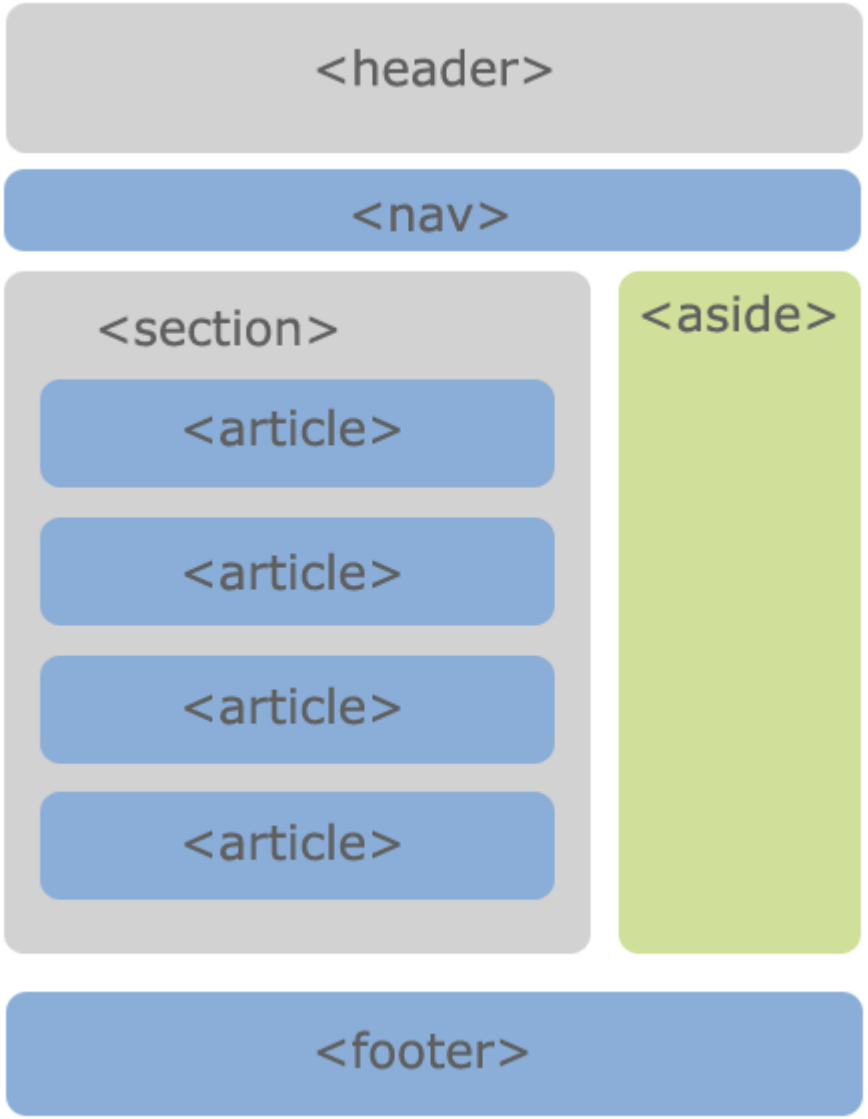
- <header>
  - Определяет «шапку» сайта или раздела.
- <footer>
  - Задаёт «подвал» сайта или раздела, в нем обычно располагается имя автора, дата документа, контактная и правовая информация.
- <nav>
  - Задаёт навигацию по сайту. Если на странице несколько блоков ссылок, то в <nav> обычно помещают приоритетные ссылки. Также допустимо использовать несколько тегов <nav> в документе. Запрещается вкладывать <nav> внутрь <address>.
- <hgroup>
  - Используется для группирования заголовков веб-страницы или раздела.
- <section>
  - Определяет раздел документа, который может включать в себя заголовки, шапку, подвал и текст. Допускается вкладывать один тег <section> внутрь другого.



- <article>
  - Задаёт содержание сайта вроде новости, статьи, записи блога, форума или др.
- <aside>
  - Определяет блок, который не относится к основному контенту, для размещения рубрик, ссылок на архив, меток и другой информации. Такой блок, если он располагается сбоку, называется, как правило, «сайдбар» или «боковая панель».
- <figure>
  - Используется для группирования любых элементов, например, изображений и подписей к ним.
- <figcaption>
  - Содержит описание для тега <figure>. Тег <figcaption> должен быть первым или последним элементом в группе.
- <time>
  - Помечает текст внутри тега <time> как дата, время или одновременно дата и время. Может указываться непосредственно внутри контейнера <time>, либо задаваться через атрибут datetime.



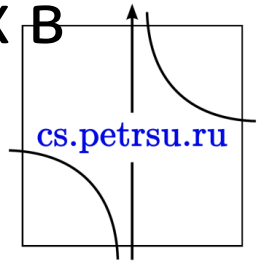
Семантическая верстка



Семантическая вёрстка, или семантический HTML-код

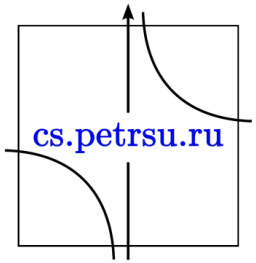
— это подход к созданию веб-страниц на языке HTML, основанный на использовании HTML-тегов в соответствии с их семантикой (предназначением), а также предполагающий логичную и последовательную иерархию страницы.

- Противопоставляется подходу, при котором написание HTML-кода определяется внешним видом веб-страницы.
- Для оформления веб-страниц, написанных в соответствии с семантикой, используются каскадные таблицы стилей (CSS).



Примеры

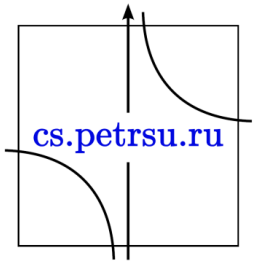
- Тег физического форматирования <i> (англ. italic, делающего текст курсивным) — вместо него теперь рекомендуется использовать тег логического форматирования (от англ. emphasis, акцентирование).
- Тег цитат <cite>
- Теги header, footer, article, aside и т.д.



CSS display table

- CSS-свойство display: table и другие, делают вывод группы элементов подобно таблице <table>
- Ограничение – объединения ячеек colspan и rowspan не поддерживаются.

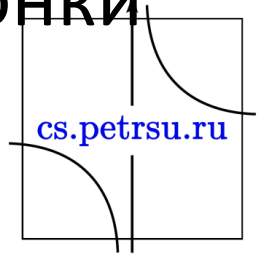
- http://htmlbook.ru/css/display



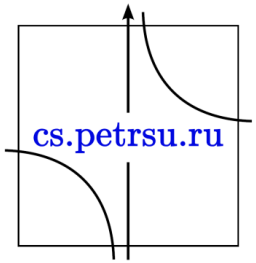
display: table;	<table>
display: inline-table;	<table> с display: inline-block
display: table-caption;	<caption>
display: table-column-group;	<colgroup>
display: table-column;	<col>
display: table-header-group;	<thead>
display: table-row-group;	<tbody>
display: table-footer-group;	<tfoot>
display: table-row;	<tr>
display: table-cell;	<td> или <th>

- Табличная верстка используется в качестве сеточной разметки
- Могут возникнут проблемы у мобильной или адаптивной версии.
- Для сеток специально ввели CSS Grid Layout

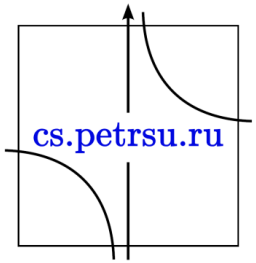
Рассмотрим пример – сетка 3x2 в три колонки



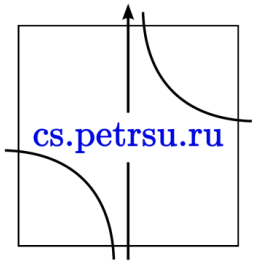
```
<table class="grid">
        <tr>
                <td><div class="item">Блок №1</div></td>
                <td><div class="item">Блок №2</div></td>
                <td><div class="item">Блок №3</div></td>
        </tr>
        <tr>
                <td><div class="item">Блок №4</div></td>
                <td><div class="item">Блок №5</div></td>
                <td><div class="item">Блок №6</div></td>
        </tr>
</table>
```



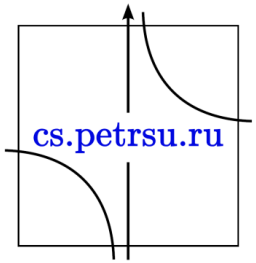
```
.grid {
        width: 100%;
        border: none;
        border-collapse: separate;
        border-spacing: 5px;
}
.grid td {
        width: 33.3%;
        vertical-align: top;
        padding: 0;
}
.item {
        background: #e07272;
        color: #fff;
        text-align: center;
        margin: 0;
        padding: 20px 0;
}
```



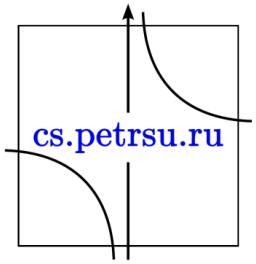
Блок №1	Блок №2	Блок №3
Блок №4	Блок №5	Блок №6



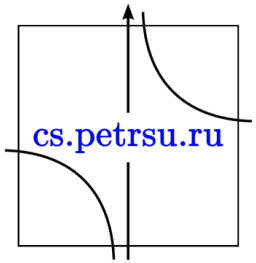
```
<div class="grid">
    <div class="grid-tr">
        <div class="grid-td"><div class="item">Блок №1</div></div>
        <div class="grid-td"><div class="item">Блок №2</div></div>
        <div class="grid-td"><div class="item">Блок №3</div></div>
    </div>
    <div class="grid-tr">
        <div class="grid-td"><div class="item">Блок №4</div></div>
        <div class="grid-td"><div class="item">Блок №5</div></div>
        <div class="grid-td"><div class="item">Блок №6</div></div>
    </div>
</div>
```



```
.grid {
        display: table;
        width: 100%;
        border: none;
        border-collapse: separate;
        border-spacing: 5px;
}
.grid-tr {
         display: table-row;
}
.grid-td {
        display: table-cell;
        width: 33.3%;
        vertical-align: top;
        padding: 0;
}
.item {
        background: #e07272;
        color: #fff;
        text-align: center;
        margin: 0;
        padding: 20px 0;
}
```

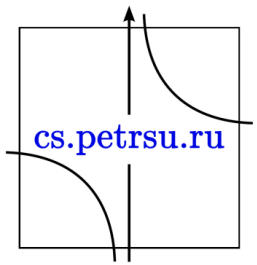


Блок №1	Блок №2	Блок №3
Блок №4	Блок №5	Блок №6



Плюсы

- Динамическое вертикальное выравнивание по центру
- Динамическое горизонтальное выравнивание по центру
- Адаптивная вёрстка
- Прилипающий подвал

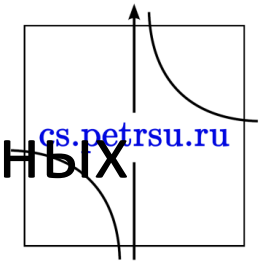


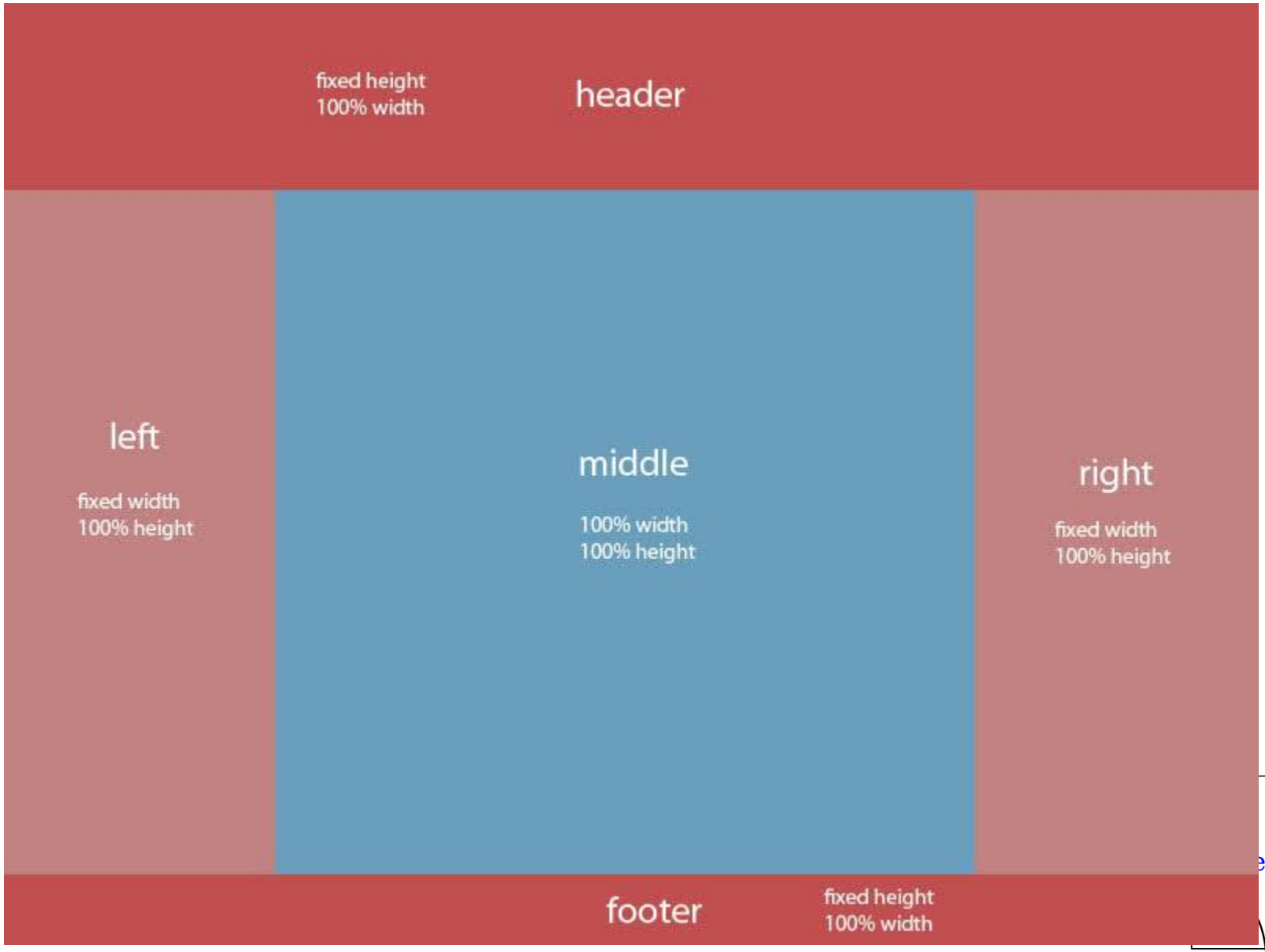
Разметка «Holy Grail» или Священный Грааль

— это вариант разметки страницы с шапкой, тремя равными по высоте колонками (две фиксированных боковых колонки и тянущийся центр) и прилипающим подвалом.

Должна удовлетворять следующим требованиям:

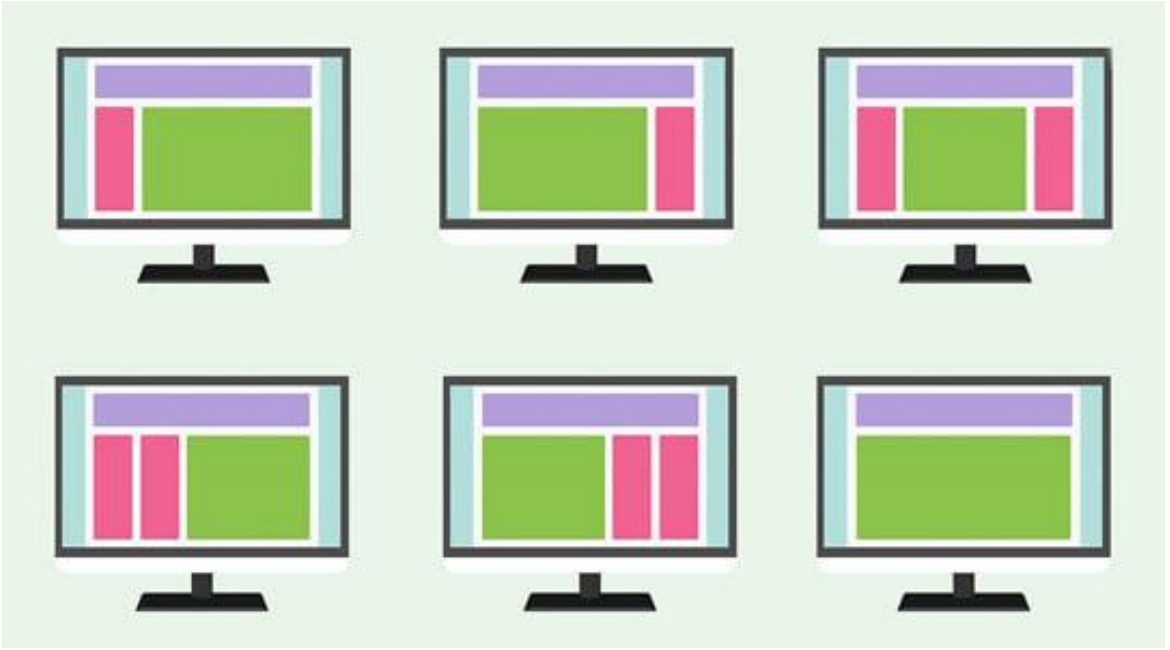
- Центральная колонка должна тянуться, а боковые — иметь фиксированную ширину.
- Центральная колонка может идти первой в разметке.
- Любая колонка может быть больше остальных по высоте.




fixed height
100% width
header
left
fixed width
100% height
middle
100% width
100% height
right
fixed width
100% height
footer
fixed height
100% width

Колоночные шаблоны

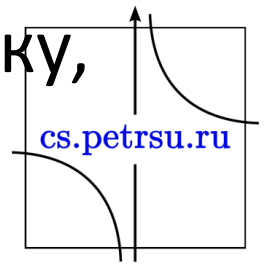
- Шаблон с одной колонкой
- Колонка слева
- Колонка справа
- Три колонки



CSS Grid Layout

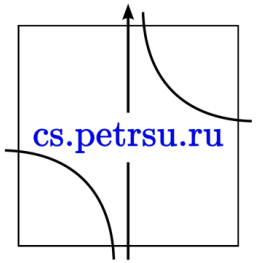
— представляет двумерную сетку для CSS.

- Grid можно использовать для размещения основных областей страницы или небольших элементов пользовательского интерфейса.
- Grid представляет собой пересекающийся набор горизонтальных и вертикальных линий - один набор определяет столбцы, а другой строки.
- Элементы Grid могут быть помещены в сетку, соответственно строкам и столбцам.



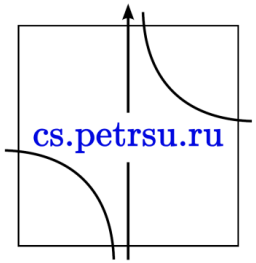
Фиксированные и гибкие размеры

- Можно создать сетку с фиксированными размерами полос — например, используя пиксели.
- Можно создать сетку с гибкими размерами, используя проценты или новую единицу измерения - fr, предназначенную для этой цели.



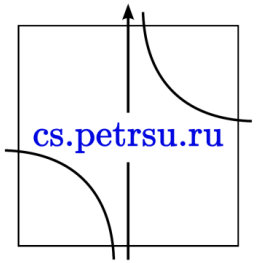
Расположение элемента

- Можно размещать элементы в заданном месте Grid, используя номера строк, имена или путём привязки к области Grid.
- Grid также содержит алгоритм управления размещением элементов, не имеющих явной позиции в Grid.



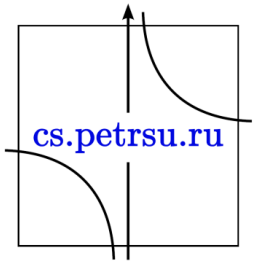
Управление выравниванием

- Grid содержит функции выравнивания, чтобы мы могли контролировать, как элементы выравниваются после размещения в области сетки, и как выравнивается весь Grid.

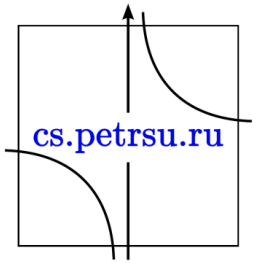


Управление перекрывающимся контентом

- В ячейку сетки может быть помещено более одного элемента, или области могут частично перекрывать друг друга.
- Это расслоение можно контролировать с помощью z-index.

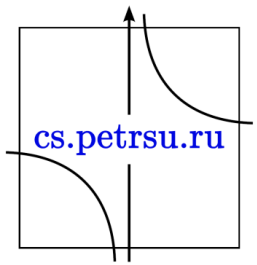


- Grid - это мощная спецификация и в сочетании с другими частями CSS, такими как flexbox, может помочь вам создать макеты, которые ранее невозможно было построить в CSS.
- Использование Grid начинается с создания сетки в вашем grid контейнере:



Grid контейнер

- Мы создаём grid контейнер, объявляя display: grid или display: inline-grid на элементе.
- Как только мы это сделаем, все прямые дети этого элемента станут элементами сетки.

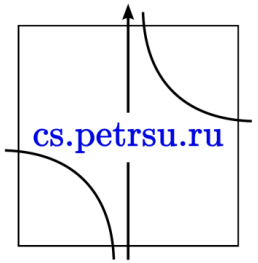


HTML

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

CSS

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



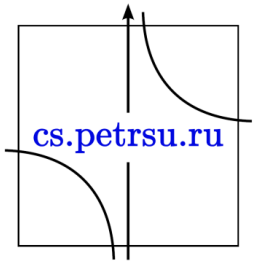
One

Two

Three

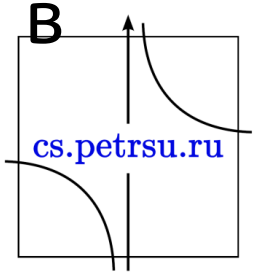
Four

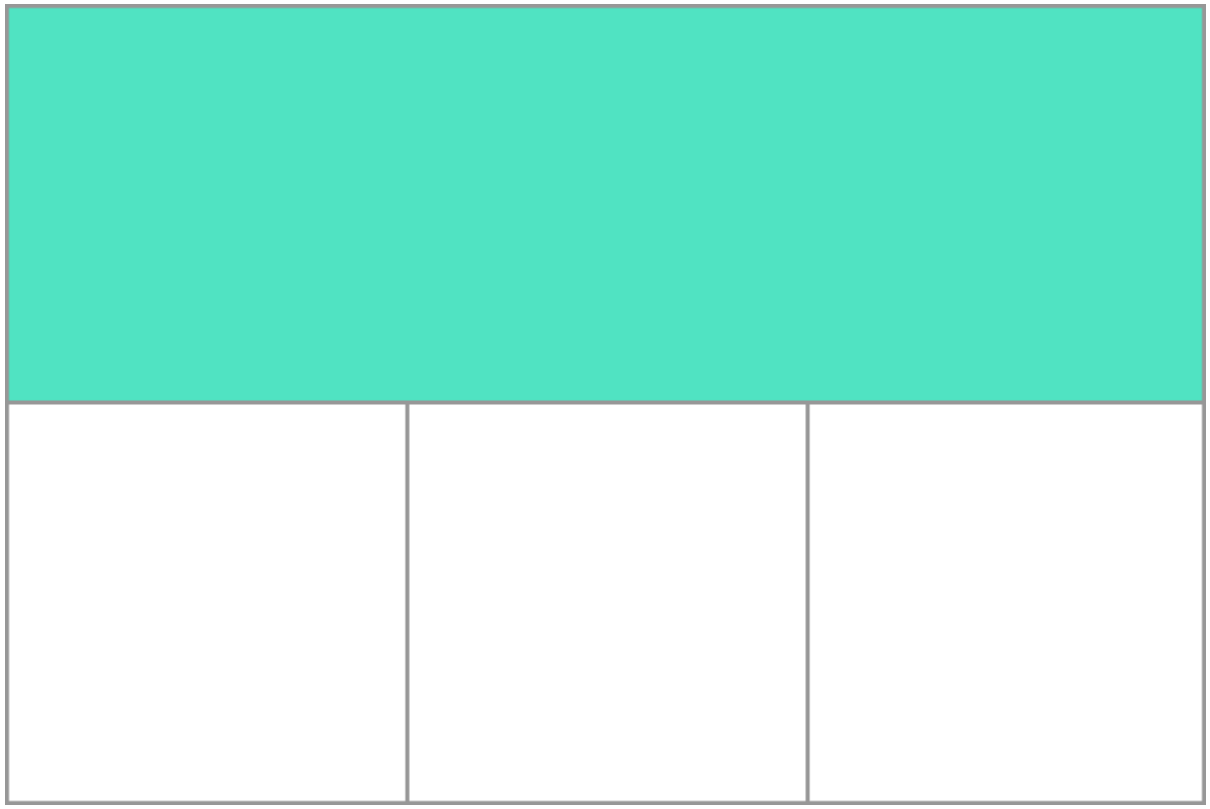
Five



Grid Tracks

- Определяем ряды и столбцы в нашей сетке при помощи свойств grid-template-columns и grid-template-rows.
- Это задает полосы сетки.
- Полоса сетки — это место между любыми двумя линиями сетки.
- На следующем слайде можно увидеть подсветку полосы — первого трека ряда в нашей сетке.

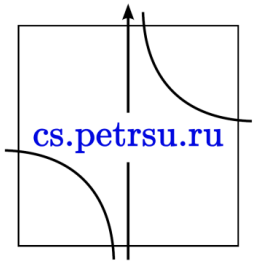




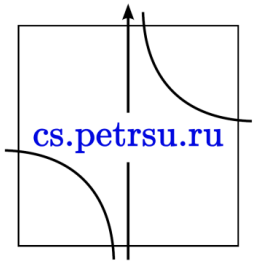
Внесем изменения:

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

```
.wrapper {
  display: grid;
  grid-template-columns:
200px 200px 200px;
}
```

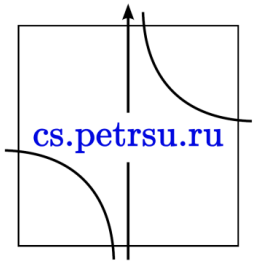


One	Two	Three
Four	Five	



Единица измерения fr

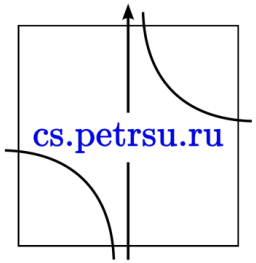
- Размер треков (полос) может быть задан с помощью любой единицы длины.
- Спецификация также вводит дополнительную единицу длины, позволяющую создавать эластичные (flexible) грид-треки.
- Новая единица длины fr представляет собой долю (fraction) доступного пространства в грид-контейнере.
- Определение грида в следующем примере создаст три трека равной длины, которые будут увеличиваться и уменьшаться в размерах в соответствии с доступным пространством.



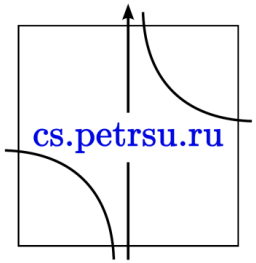
Внесем изменения:

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

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

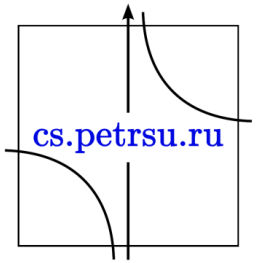


One	Two	Three
Four	Five	



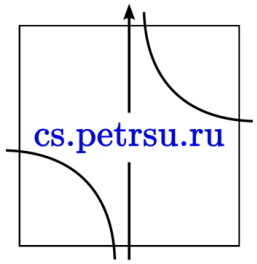
- В следующем примере мы создаём грид с треком в 2fr и двумя треками по 1fr.
- Доступное пространство разбивается на четыре части.
- Две части занимает первый трек, и две части - два оставшихся.

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



- В последнем примере смешаем треки с абсолютной длиной и треки с длиной во fr.
- Размер первого трека 500 пикселей, и эта фиксированная ширина убирается из доступного пространства.
- Оставшееся пространство разбивается на три части и пропорционально разделяется между двумя эластичными треками.

```
.wrapper {
  display: grid;
  grid-template-columns: 500px 1fr 2fr;
}
```

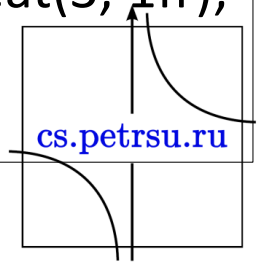


Нотация repeat()

- В огромных гридах с большим количеством треков можно использовать нотацию repeat() , чтобы повторить структуру треков или её часть.
- Например, такое задание грида:

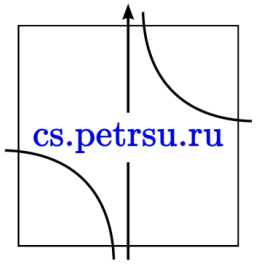
```
.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```



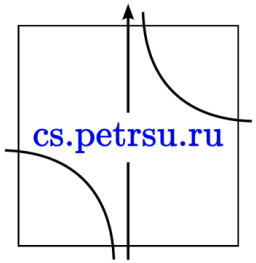
- Repeat-нотацию можно использовать как часть списка треков.
- В следующем примере создаётся грид с начальным треком шириной в 20 пикселей, шестью треками шириной в 1fr и последним треком шириной в 20 пикселей.

```
.wrapper {
  display: grid;
  grid-template-columns: 20px repeat(6, 1fr) 20px;
}
```



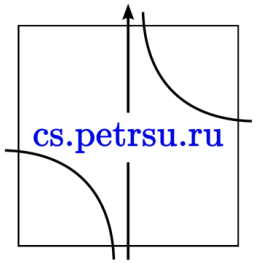
Явный и неявный грид

- Во всех примерах выше мы создавали наши колоночные (столбцовые) треки с помощью свойства grid-template-columns, в то время как грид самостоятельно создавал строки (ряды, полосы) для любого контента там, где это требовалось.
- Эти строки создавались в неявном гриде.



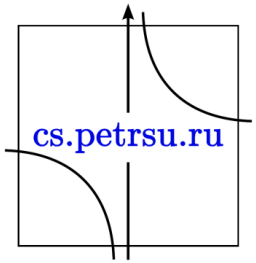
Явный и неявный грид

- Явный грид состоит из строк и колонок, которые мы определяем с помощью grid-template-columns и grid-template-rows. Если вы размещаете что-нибудь вне рамок определённого данными свойствами грида или в зависимости от контента требуется большее количество грид-треков, грид создаёт строки и колонки в виде неявного грида.
- Размер подобных треков по умолчанию задаётся автоматически в зависимости от находящегося в них контента.



- Можно задать размер треков, создаваемых в виде неявного грида с помощью свойств grid-auto-rows и grid-auto-columns.

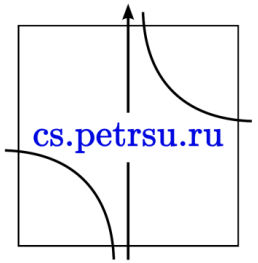
```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 200px;
}
```



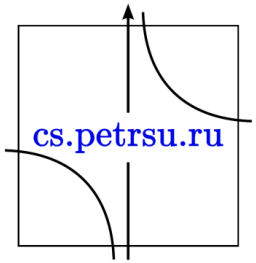
Масштабирование треков

- Определение для треков минимального размера и растяжение треков под контент.
  - Например, нам нужно, чтобы строки никогда не становились меньше 100 пикселей, но если контент занимает, скажем, 300 пикселей в высоту, мы хотим, чтобы строка тоже стала 300 пикселей.

- Для подобных ситуаций в Grid предусмотрено решение с помощью функции minmax().
  - grid-auto-rows: minmax(100px, auto);

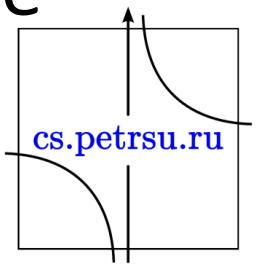


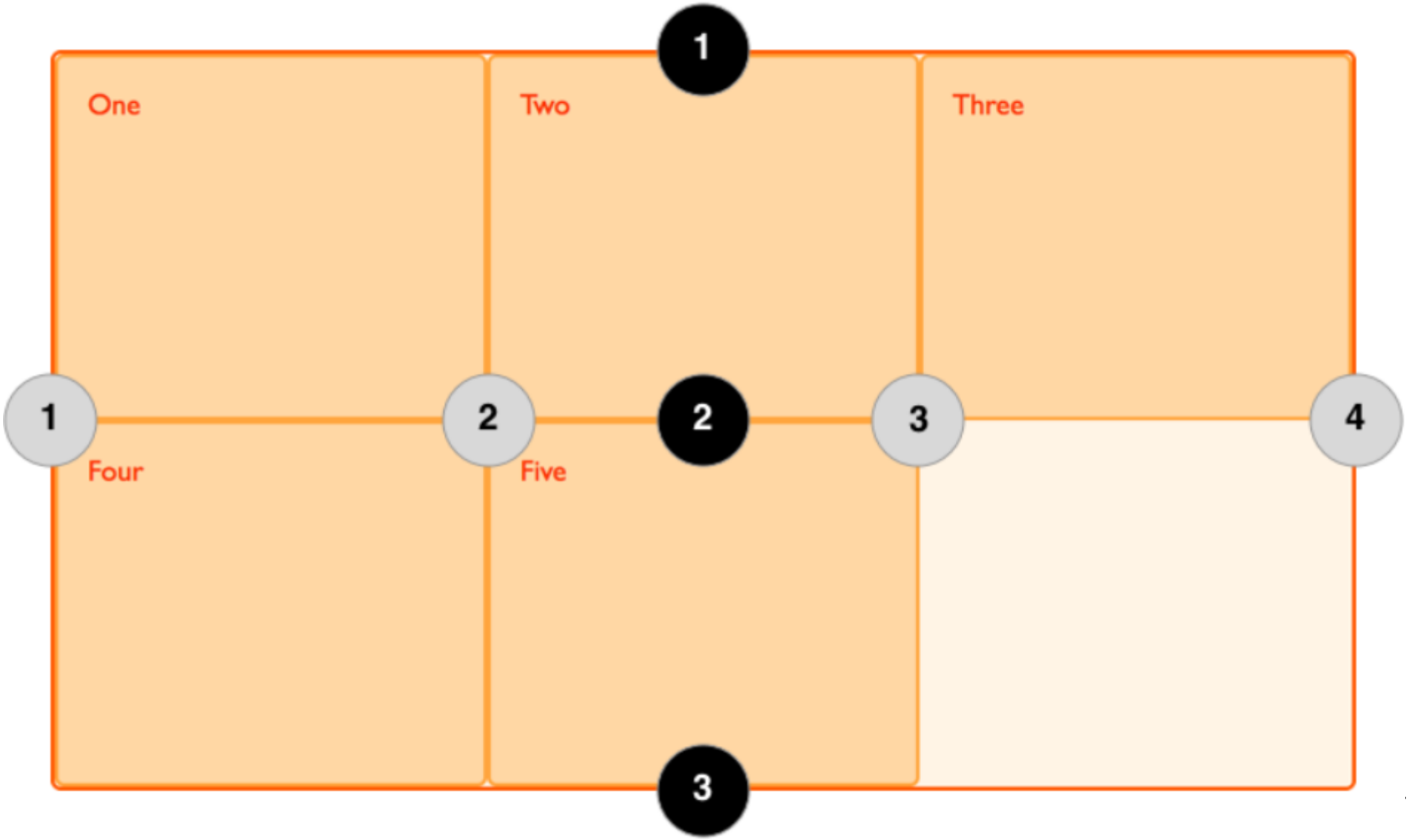
One	Two I have some more content in. This makes me taller than 100 pixels.	Three
Four	Five	



Grid-линии

- Когда мы определяем грид, мы определяем грид-треки, а не грид-линии.
- После этого грид обеспечивает нас пронумерованными линиями, номера которых можно использовать для размещения элементов.
- Например, в гриде с тремя колонками и двумя рядами у нас - четыре колоночные линии.

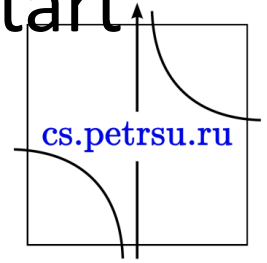



1
One
Two
Three
1
2
2
3
4
Four
Five
3

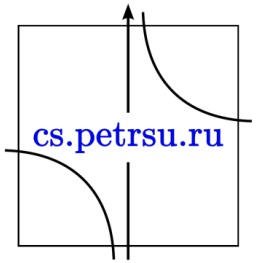
Размещение элементов по линиям

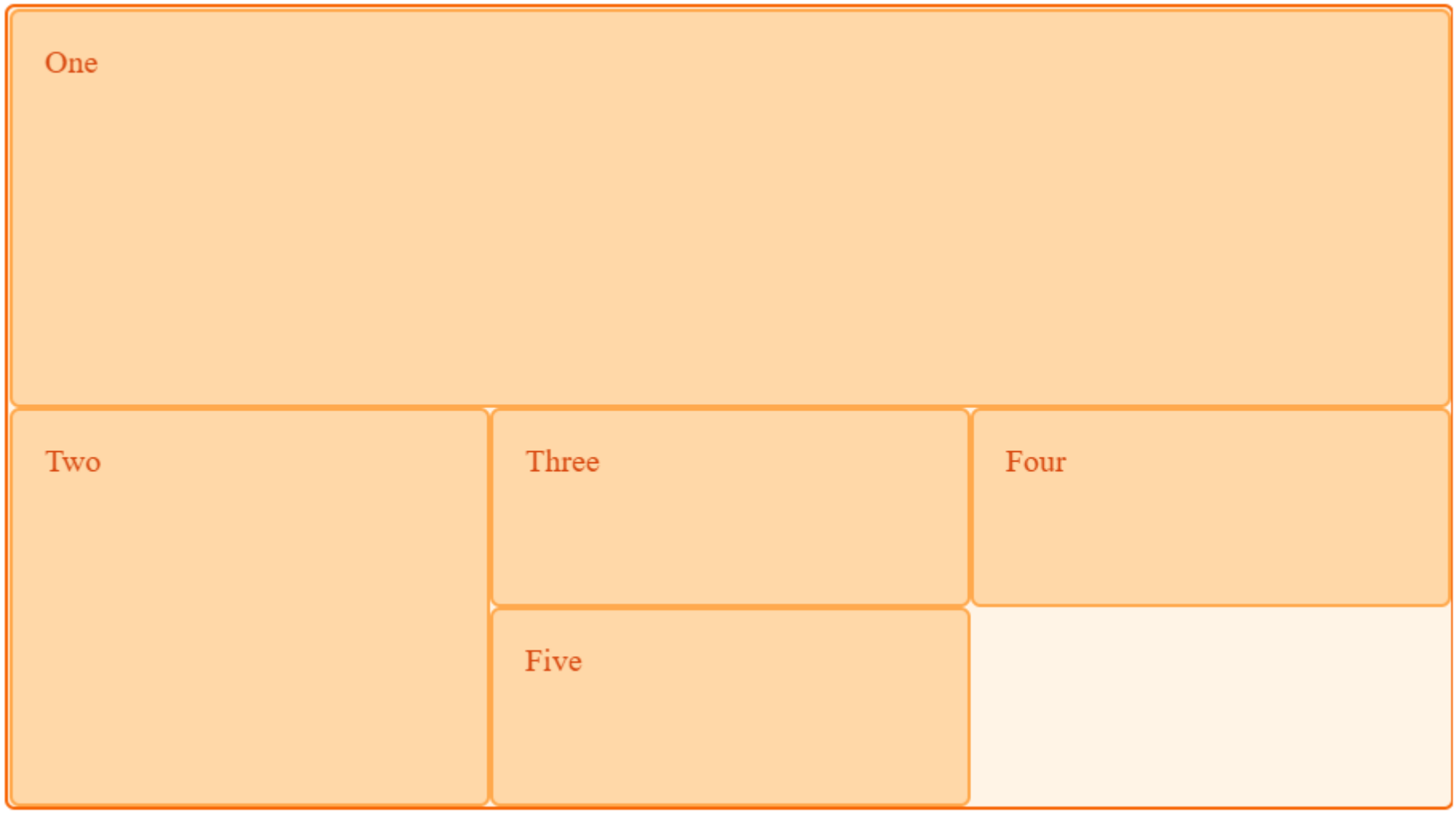
- Рассмотрим простой способ.
- При размещении элемента мы ссылаемся именно на линию, а не на трек.

- В следующем примере мы размещаем первые два элемента в нашем гриде из трёх колоночных треков с помощью свойств grid-column-start, grid-column-end, grid-row-start и grid-row-end.



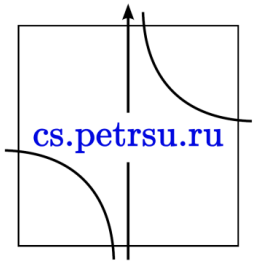
```
.wrapper {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-auto-rows: 100px;
}
.box1 {
    grid-column-start: 1;
    grid-column-end: 4;
    grid-row-start: 1;
    grid-row-end: 3;
}
.box2 {
    grid-column-start: 1;
    grid-row-start: 3;
    grid-row-end: 5;
}
```

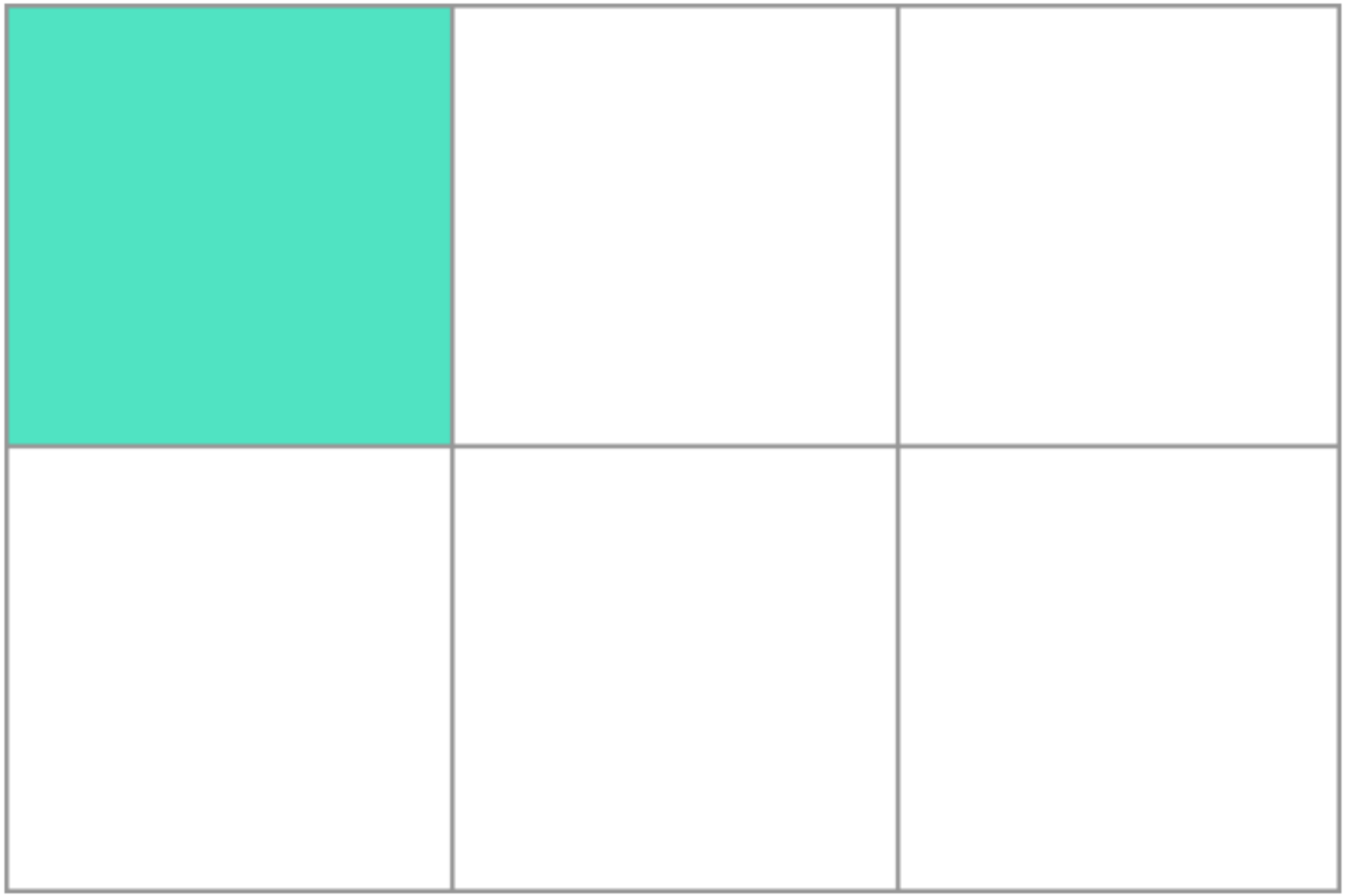




Grid-ячейки

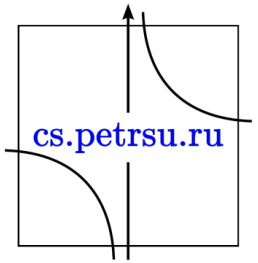
- Грид-ячейка (grid cell) - наименьшая единица измерения грида, концептуально похожая на ячейку таблицы.
- Как мы видели в предыдущих примерах, едва грид определён для родительского элемента, дочерние элементы автоматически размещаются в каждой ячейке нашего заданного грида.
- На рисунке ниже первая ячейка грида выделена цветом.

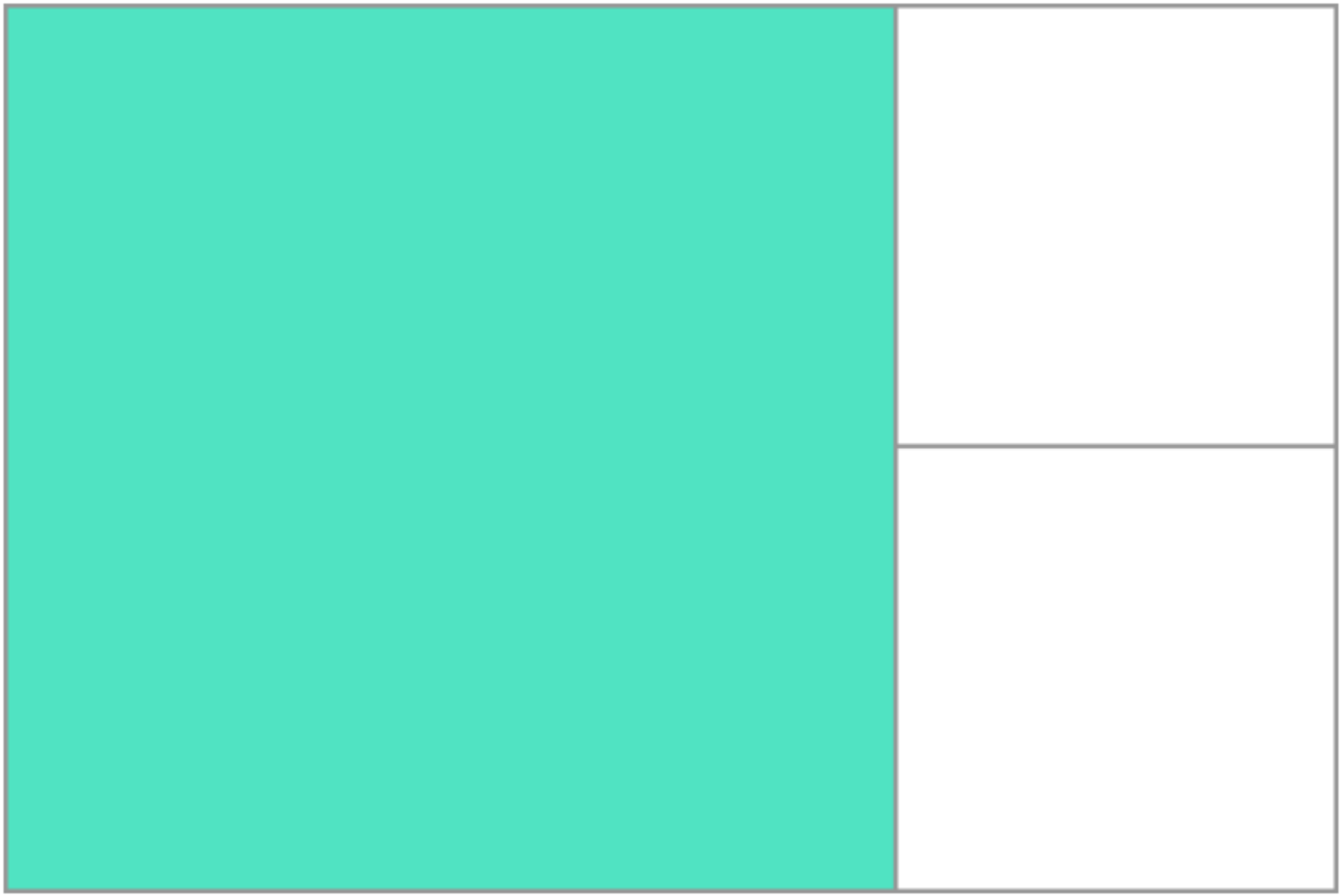




Grid-области

- Элементы могут занимать одну или несколько ячеек внутри строки или колонки, и подобное поведение создаёт грид-область (grid area).
- Грид-области должны быть перпендикулярными, - невозможно создать область, например, в форме буквы L.
- Выделенная цветом грид-область на рисунке ниже занимает два строчных трека и два колоночных.

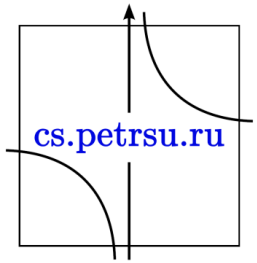




Зазоры (Gutters)

- Зазоры (Gutters) или аллеи (alleys) между грид-ячейками можно создать с помощью свойств grid-column-gap и grid-row-gap, или с помощью сокращённого свойства grid-gap.
- В примере ниже мы создаём зазор в 10 пикселей между колонками и в 1em между строками.

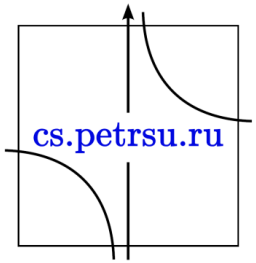
```
grid-column-gap: 10px;
grid-row-gap: 1em;
```



- Любое пространство, которое занимают зазоры, добавляется в начало эластичных треков, задаваемых с помощью fr, поэтому зазоры используются для регулирования размеров и действуют как регулярные грид-треки, хотя что-то разместить в них нельзя.
- В терминах расположения элементов по грид-линиям (line-based positioning) зазоры ведут себя так, как если бы самой грид-линии была добавлена толщина.

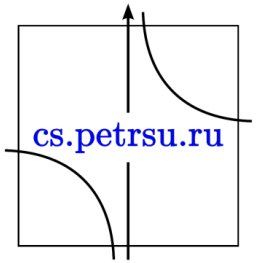
Вложенные гриды

- Грид-элемент может быть и грид-контейнером.
- В следующем примере у нас есть созданный ранее трёхколоночный грид, в котором размещены два элемента.
- В данном случае у первого элемента есть несколько подэлементов.
- Поскольку эти подэлементы не являются прямыми потомками грида, они не участвуют в структуре грида и отображаются в соответствии с нормальным потоком документа.

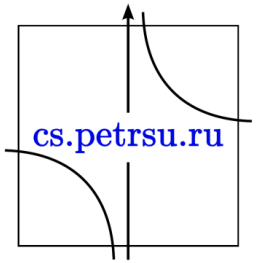


```
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

```
<div class="wrapper">
  <div class="box box1">
    <div class="nested">a</div>
    <div class="nested">b</div>
    <div class="nested">c</div>
  </div>
  <div class="box box2">Two</div>
  <div class="box box3">Three</div>
  <div class="box box4">Four</div>
  <div class="box box5">Five</div>
</div>
```



a	b	c
Two		
Three		
Four		
Five		

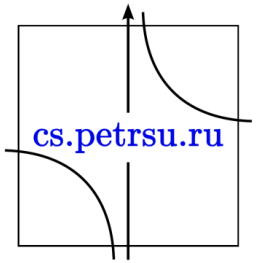


CSS flexbox

(Flexible Box Layout Module)

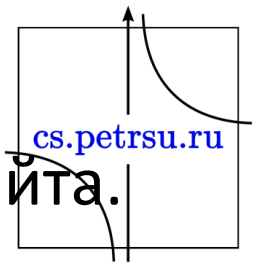
— модуль макета гибкого контейнера — представляет собой способ компоновки элементов, в основе лежит идея оси.

- Flexbox состоит из гибкого контейнера (flex container) и гибких элементов (flex items).
- Гибкие элементы могут выстраиваться в строку или столбик, а оставшееся свободное пространство распределяется между ними различными способами.



Позволяет решать задачи:

- Располагать элементы в одном из четырех направлений: слева направо, справа налево, сверху вниз или снизу вверх.
- Переопределять порядок отображения элементов.
- Автоматически определять размеры элементов таким образом, чтобы они вписывались в доступное пространство.
- Решать проблему с горизонтальным и вертикальным центрированием.
- Переносить элементы внутри контейнера, не допуская его переполнения.
- Создавать колонки одинаковой высоты.
- Создавать прижатый к низу страницы подвал сайта.



Наглядное введение в систему компоновки элементов на веб-странице

- https://tproger.ru/translations/how-css-flexbox-works/