HyperText Markup Language

HTML&CSS

Урок 8

Модель Flexbox

Оглавление

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Модель Flexbox

Долгое время верстальщики мечтали об инструменте, позволяющем им размещать по 2, 3, 4 и более колонки в один ряд очень простым способом. Сначала это было реализовано с помощью таблиц, затем с помощью свойства float: left или display: inline-block, но каждый из этих способов имел свои нюансы и ограничения. Поэтому w3.org в 2009 был опубликован черновик «Flexible Box Layout Module», в 2011 и 2012 году он претерпел изменения, и наконец, с 2014 года в стандарт CSS3 вошел в том виде, в котором мы используем его в данный момент.

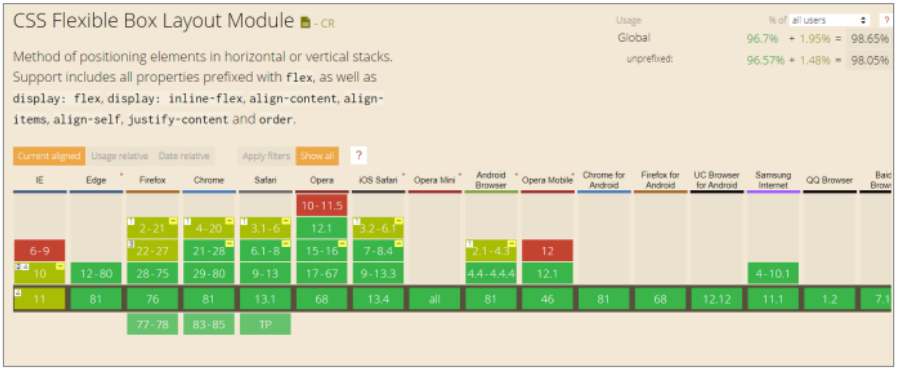


Рисунок 1

Сейчас Flexbox уже прочно обосновался в арсенале верстальщиков и широко используется повсеместно, в том числе и в таких фреймворках, как Bootstrap-4 или Foundation-6. К его плюсам можно отнести очень хорошую поддержку всеми современными браузерами и частичную — теми версиями браузеров, которые были

выпущены между 2011 и 2013-м годом. Для последних необходимо указывать ряд свойств с префиксами -webkit- (*Chrome, Safari*), -ms- (*Internet Explorer10*), -moz- (*Mozilla Firefox*). Более подробную информацию по этому вопросу вы найдете на сайте caniuse.com (рис. 1).

К условным минусам Flexbox можно отнести отсутствие поддержки в Internet Explorer версий 6-9, которыми на данный момент пользуется порядка 0,33% всех пользователей Интернета и наличие нескольких версий написания свойств для тех браузеров, которые были выпущены до 2014 года. Такими браузерами пользуется на момент написания урока порядка 0,5% от всех пользователей Интернета в мире. Кроме того, есть ряд багов Flexbox-модели, которые актуальны для ряда браузеров, особенно для Internet Explorer версий 10-11. Также к минусам Flexbox модели можно отнести то, что часть значений для ряда свойств описаны в стандарте, но еще не поддерживаются браузерами. Хотя тех, что поддерживаются, вполне достаточно для верстки.

Ссылки на полезные ресурсы:

- https://caniuse.com/#search=flex;
- https://github.com/philipwalton/flexbugs#flexbugs.

Что такое Flexbox?

Давайте разберемся с тем, что представляет собой Flexbox модель.

Flexbox (*CSS Flexible Box Layout*) — это модель отображения содержимого страницы, при которой вы получаете набор гибких элементов внутри контейнера, имеющего свойство display со значением flex или inline-flex.

Из данного определения следует, что у нас есть один родительский элемент и несколько вложенных, или дочерних элементов. Собственно, и свойства, которые существуют в модели Flexbox, подразделяются на свойства для элемента-контейнера и для дочерних элементов. Рассмотрим их последовательно.

Предположим, что у нас есть 4 элемента внутри пятого , который является их родителем (или контейнером). Каждый из этих элементов имеет цифру от 1 до 4-х — порядковый номер, обозначающий последовательность их появления в html-разметке, высоту в 100px и свой цвет фона. У каждого div-а есть свойство display: block по умолчанию, поэтому они расположены друг под другом и занимают все свободное пространство внутри родительского div-а, *обведенного серой рамкой*.

Рисунок 2

С помощью свойств Flexbox модели мы можем изменить положение дочерних элементов внутри роди-

тельского контейнера, применив ряд соответствующих свойств.

Свойства flex-контейнера

Элемент, который содержит ряд вложенных элементов, мы будем называть flex-контейнером. Для него обязательно нужно задать свойство display:

```
display: flex | inline-flex
```

Разница между двумя значениями заключается в том, что при display: flex размер контейнера по ширине равен всему доступному пространству внутри родительского элемента (аналогично display: block), а при display: inline-flex размер контейнера определяется размером вложенных элементов (аналогично display: inline-block). Обратите внимание на черную рамку, которая на первой картинке значительно длиннее, чем на второй.

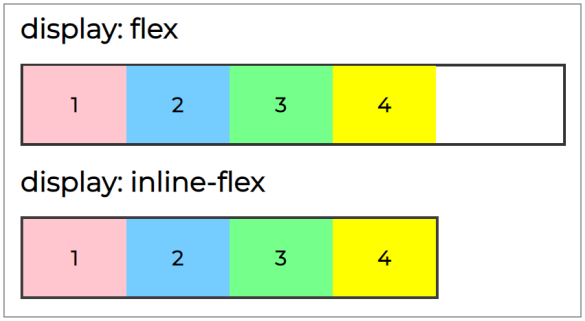


Рисунок 3

Главная и поперечная ось

Как видно из рисунка выше, все элементы flex-контейнера выстроились в одну горизонтальную линию. Такое

поведение задается еще одним свойством для flex-контейнера — flex-direction:

```
flex-direction: row | row-reverse | column | column-reverse
```

На рисунке видно, что вложенные элементы перестраиваются в зависимости от значения. Направление row или row-reverse выстраивает дочерние элементы по горизонтали в направлении слева направо или справа налево. Значения column или column-reverse меняют направление размещения блоков на вертикальное: сверху вниз или снизу вверх.

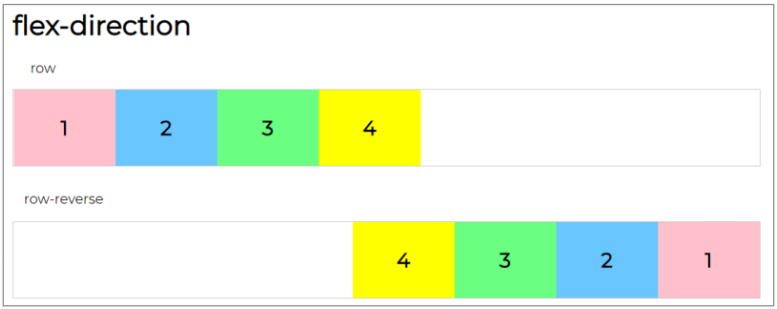


Рисунок 4

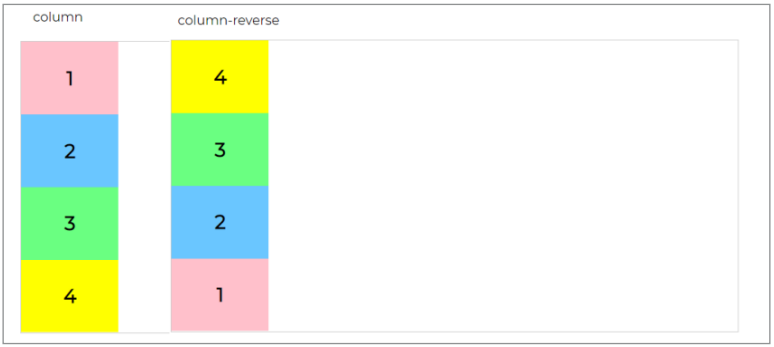


Рисунок 5

В спецификации CSS для Flex-контейнера существует понятие главной и поперечной оси. **Главная ось** (main axis) направлена слева направо (для арабских языков (rtl) — справа налево), а **поперечная** (cross axis) — сверху вниз.

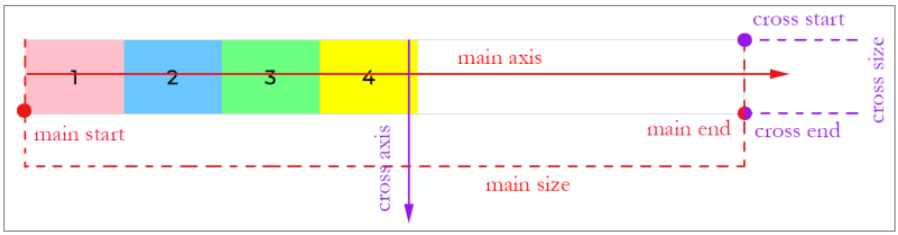


Рисунок 6

Именно **направление главной оси**, а также ее начало и конец (main start и main end) определяет размещение вложенных flex-элементов при значении flex-direction, заданных как row или row-reverse. В случае изменения направления flex-direction на column или column-reverse, определяющей становится **поперечная ось** и ее начало или конец (cross start и cross end). Размер главной (main size) или поперечной оси (cross size) важен для расчета размеров вложенных элементов, особенно, если они заданы в %.

От направления главной и поперечной оси зависят остальные свойства flex-контейнера. Мы будем из рассматривать в основном для flex-direction: row. Это значение по умолчанию, как правило, достаточно для размещения большинства flex-элементов.

Свойство justify-content

Свойство justify-content определяет выравнивание flex-элементов вдоль главной оси. Оно часто используется

при построении колонок, т.к. позволяет распределить контент внутри контейнера. Значения:

```
justify-content: flex-start | flex-end | center |
                 space-between | space-around |
                 space-evenly
```

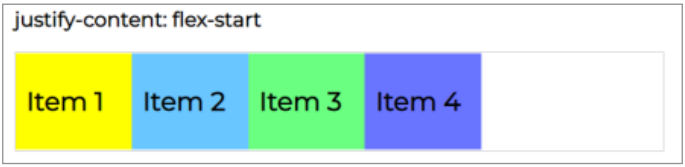


Рисунок 7

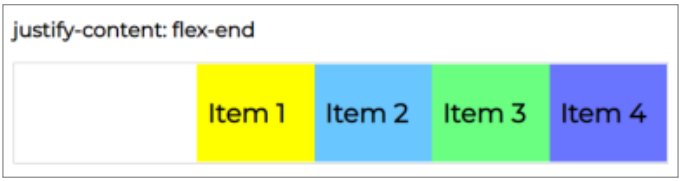


Рисунок 8

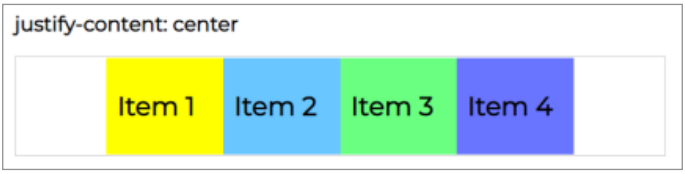


Рисунок 9

Рисунок 10

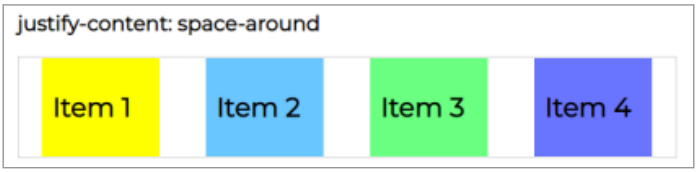


Рисунок 11

Рисунок 12

Свойство align-items

Свойство align-items определяет, каким образом выравнивать flex-элементы **вдоль поперечной оси**. Для корректного использования этого свойства необходимо задать высоту контейнера больше, чем высота вложенных элементов (часто бывает в header). Значения:

```
align-items: flex-start | flex-end | center | stretch |
             baseline
```

Рисунок 13

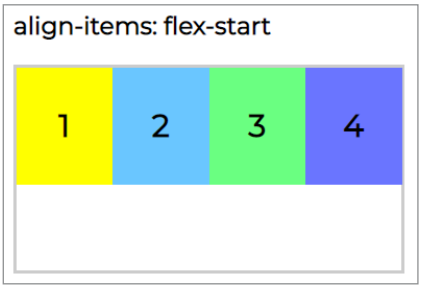


Рисунок 14

Рисунок 15

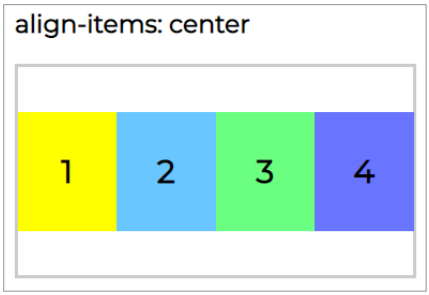


Рисунок 16

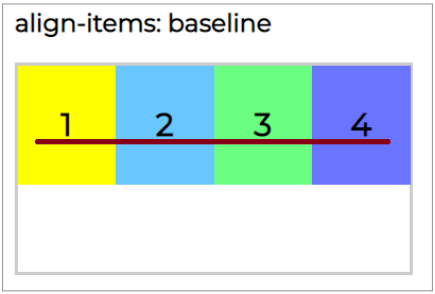


Рисунок 17

Многострочность внутри flex-контейнера

Далеко не всегда располагать элементы нужно только в одну строку. Поэтому в стандарте CSS предусмотрены свойства для многострочного flex-контейнера.

Свойство flex-wrap

Свойство flex-wrap отвечает за то, как будут размещаться вложенные flex-элементы внутри контейнера, если они не помещаются в одну строку. То есть это свойство управляет переносами строк внутри flex-контейнера.

```
flex-wrap: nowrap | wrap | wrap-reverse
```

Рисунок 18

При назначении flex-wrap в виде значения wrap элементы переносятся на новые строки в том порядке, в котором они расположены в html-разметке. Значение wrap-reverse оборачивает этот порядок, располагая первые дочерние элементы внизу, а последние — в верхнем ряду. Размер и количество строк во flex-контейнере будут также зависеть от значений свойств width или flex-basis для дочерних flex-элементов, которое мы рассмотрим чуть ниже.

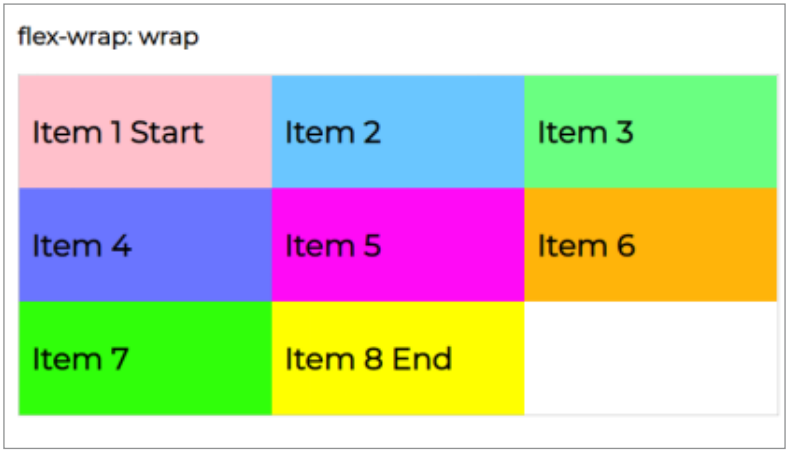


Рисунок 19

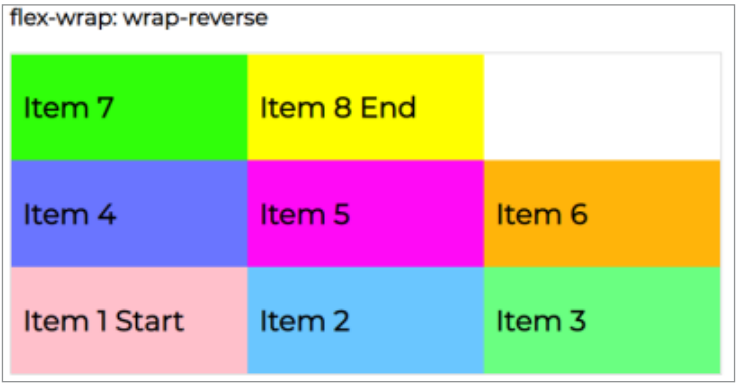


Рисунок 20

Объединенное свойство flex-flow

Свойство flex-flow предназначено для объединения свойств flex-direction и flex-wrap. По умолчанию имеет значение flex-flow: row nowrap. Имеет смысл использовать его при переопределении обоих этих значений или одного из них.

```
flex-flow: flex-direction || flex-wrap
```

Свойство align-content

Свойство align-content используется только в многострочном режиме (т.е. в случае, когда свойство flex-wrap имеет значение wrap или wrap-reverse). Оно определяет, каким образом ряды flex-элементы будут выровнены по вертикали, а также то, как они будут делить между собой все пространство flex-контейнера. Варианты значений свойства:

```
align-content: stretch | flex-start | flex-end | center
                       | space-around | space-between
```

По умолчанию свойство align-content имеет значение stretch, поэтому все flex-элементы вытянуты по высоте flex-контейнера. При изменении значений на flex-start или flex-end, они подтягиваются к верхней или нижней границе flex-контейнера (рис. 21-24).

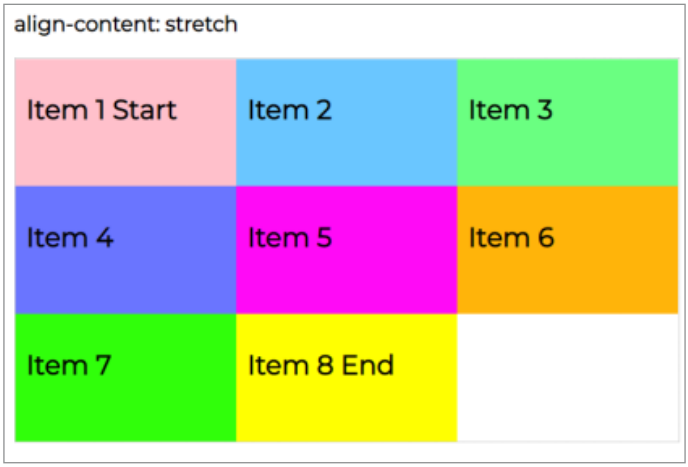


Рисунок 21

Рисунок 22

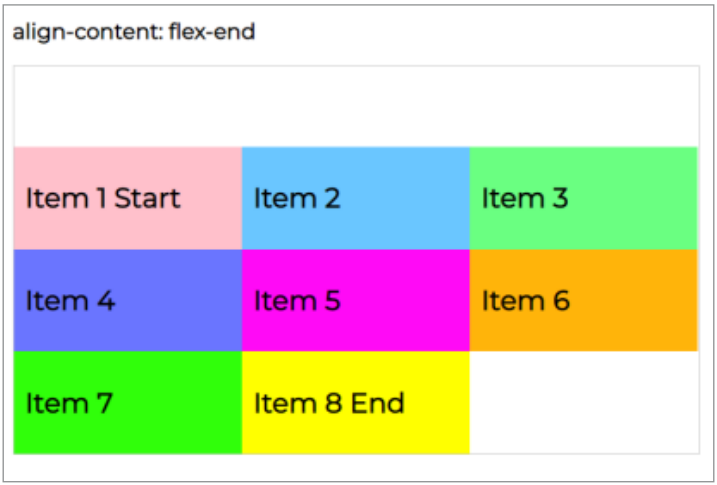


Рисунок 23

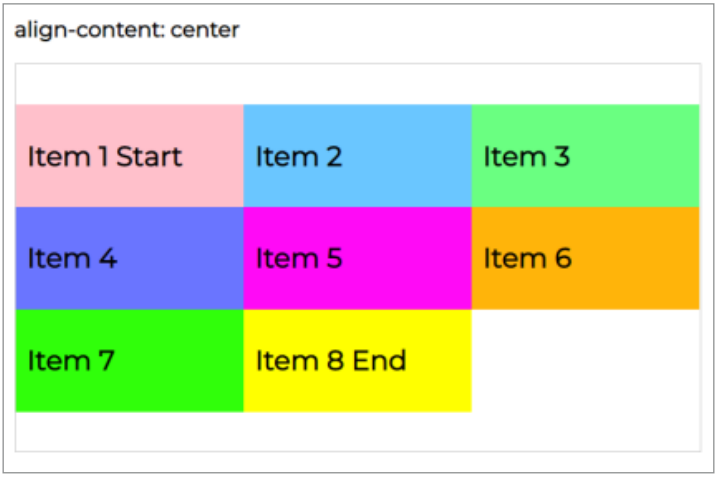


Рисунок 24

Значение свойства align-content: center располагает flex-элементы по вертикальному центру flex-контейнера, оставляя сверху и снизу равные промежутки (рис. 25-26).

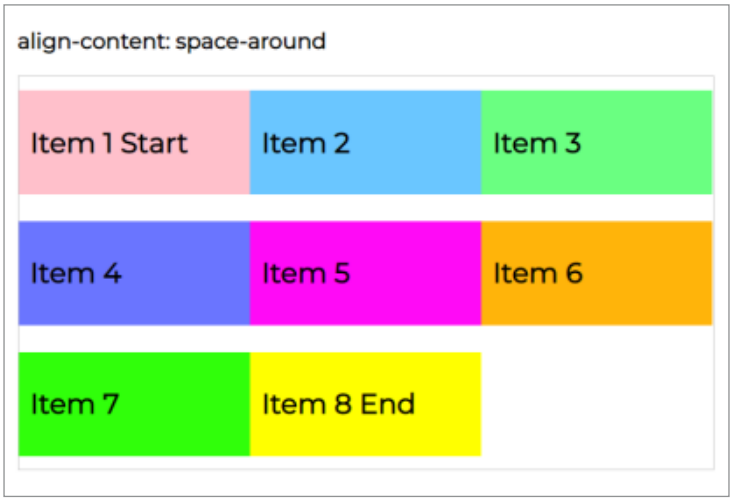


Рисунок 25

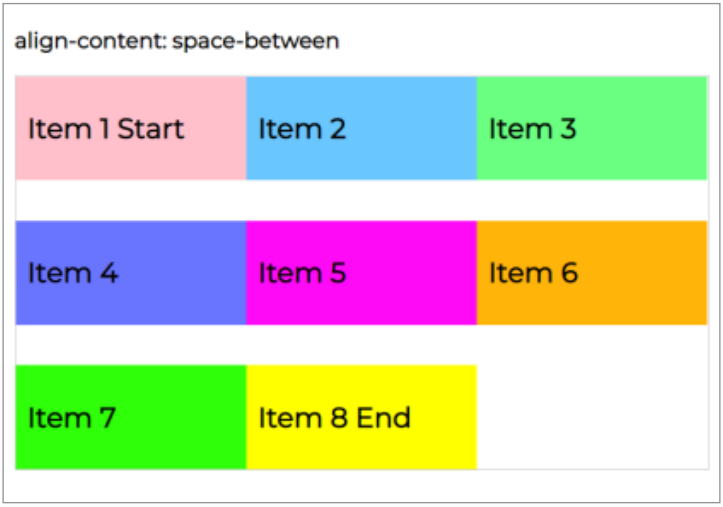


Рисунок 26

Остальные значения свойства align-content перераспределяют свободное пространство внутри контейнера так, чтобы оно делилось на равные промежутки между

всеми линиями flex-элементов (значение space-between) или с добавлением половины промежутка сверху и половины снизу (значение space-around).

Свойства flex-элементов

Свойства flex-элементов позволяют управлять каждым из элементов внутри flex-контейнера по отдельности. Это, в первую очередь, свойства, которые управляют размерами дочерних элементов, их растяжимостью и сжимаемостью, порядком отображения вне зависимости от html-разметки, а также возможностью переопределить размещение flex-элементов, заданное свойствами flex-контейнера.

Свойство flex-basis

Свойства flex-basis позволяет назначить базовый размер flex-элемента по основной оси. Вы можете задать его в любых единицах (5em, 4rem, 48%, 200px и др.) или указать, как auto (*значение по умолчанию*), но реальный размер flex-элемента будет зависеть от 2-х коэффициентов — flex-grow и flex-shrink. Также реальная ширина flex-элемента будет зависеть от min-width и max-width, назначенных для flex-элемента. Значение flex-basis будет корректироваться в зависимости от минимальной и максимальной ширины, т.е. ширина элемента не может быть меньше min-width и не может быть больше max-width. То же относится к размерам по вертикали (height, min-height, max-height), если свойство flex-direction имеет значение column (рис. 27).

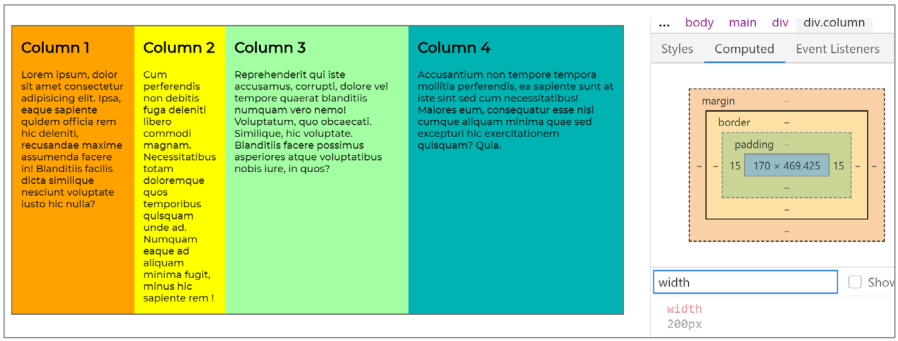


Рисунок 27

На скриншоте выше ширина flex-контейнера составляет 1000px. Первый столбец имеет flex-basis: 200px, второй — flex-basis: 10%, т.е. должен быть шириной 100px, однако текста в нем больше, поэтому реальная ширина его 149px. В 3-м столбце flex-basis: 10%, но задано еще свойство min-width: 300px, поэтому его ширина составляет 300px. Для четвертого столбца задано свойство flex-grow: 1, поэтому его ширина увеличилась до 350.875px.

Посмотреть или попробовать изменить значения свойств можно на codepen.io.

Свойство flex-grow

Свойство flex-grow определяет фактор (коэффициент) растяжимости flex-элемента при наличии избыточного пространства во flex-контейнере. По умолчанию flex-grow имеет значение 0, т.е. все flex-элементы будут того размера, который им задан с помощью width или flex-basis, и часть flex-контейнера останется незанятой (рис. 28).

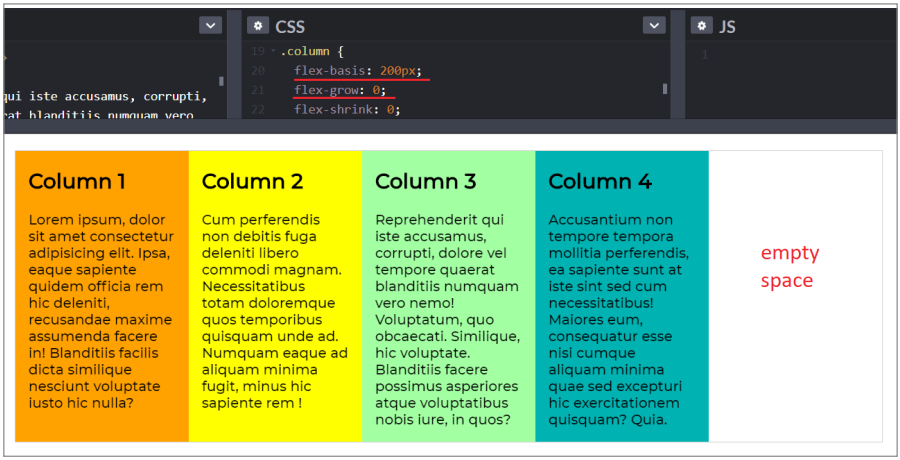


Рисунок 28

Например, flex-контейнер с шириной 1000px содержит 4 flex-элемента с шириной 200px, т.е. 200px остаются незаполненными. Если каждому из этих элементов назначить flex-grow: 1, то 200px избыточного пространства нужно разделить на сумму этих коэффициентов, т.е. 200px:4 = 50px, а потом добавить результат к ширине каждого элемента. В результате этих действий ширина каждого flex-элемента станет 250px. Пример вы найдете на codepen.io (рис. 29).

Если же какой-то один flex-элемент получит свойство flex-grow: 2, то расчет будет уже другим. 200px: 5 = 40px. Цифра 5 — это сумма flex-grow всех элементов (1+1+1+2 = 5). Затем к 3-м из наших элементов нужно добавить по 40px, и их размер станет 240px, а к тому, у которого flex-grow: 2, мы добавим 80px (40px*2) — и получим 280px. Общая сумма всех ширин будет по-прежнему 1000px (рис. 30).

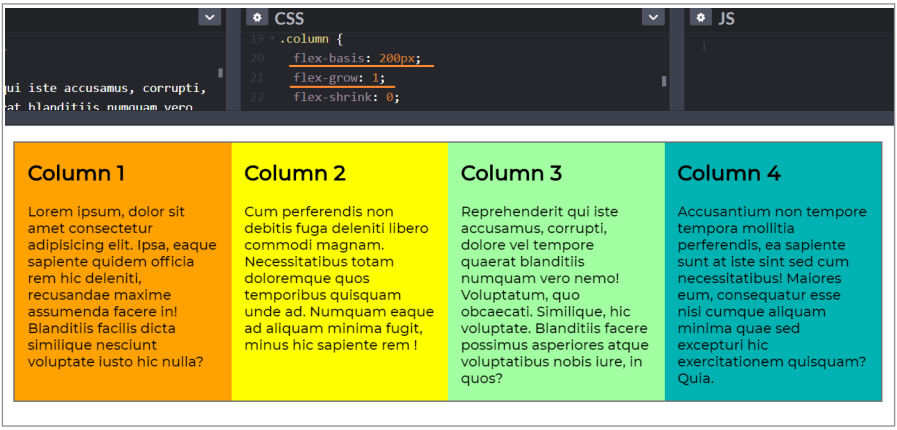


Рисунок 29

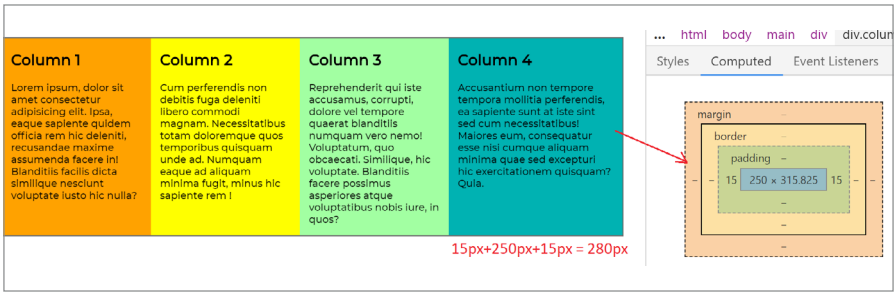


Рисунок 30

Свойство flex-shrink

Свойство flex-shrink является фактором (коэффициентом) уменьшения ширины (высоты в случае flex-direction: column) flex-элемента, если во flex-контейнере недостаточно места. По умолчанию он равен 1, т.е. ширина всех элементов будет уменьшена, если ширина контейнера недостаточна для размещения всех элементов или для каждого них задан размер, который подразумевает большую ширину flex-контейнера (рис. 31).

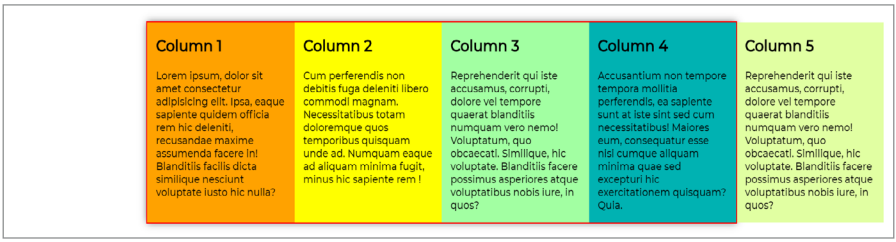


Рисунок 31

Например, ширина flex-контейнера равна 1000px, а ширина каждого из 5 flex-элементов 250px. Получится, что один из элементов должен выйти за пределы flex-контейнера. Однако, этого не происходит из-за flex-shrink: 1. Ширина каждого элемента уменьшится на 50px: 250px: 5 = 50px, и станет равна 200px: 250px-50px=200px (рис. 32).

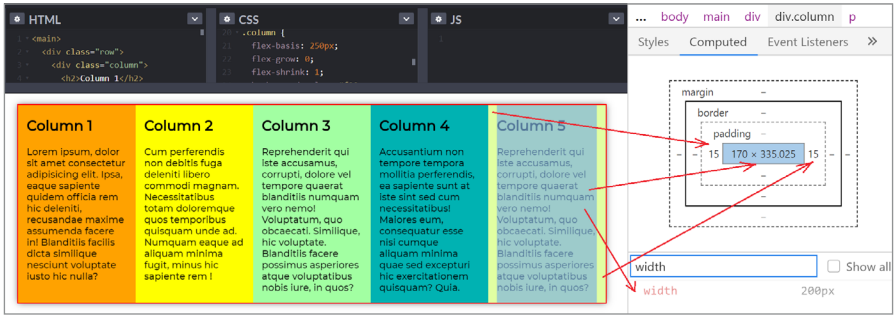


Рисунок 32

Если добавить какому-либо одному элементу flex-shrink:0, то он перестанет уменьшаться, оставаясь размером 250px. А нехватка в 250px распределится уже между 4 элементами, т.е. 250px: 4 = 62,5px. Затем браузер отнимет эту величину от ширины каждого элемента: 250px – 62.5px = 187.5px. Эта величина может быть изменена

в зависимости от контента (текста, картинок) внутри flex-элемента (рис. 33).

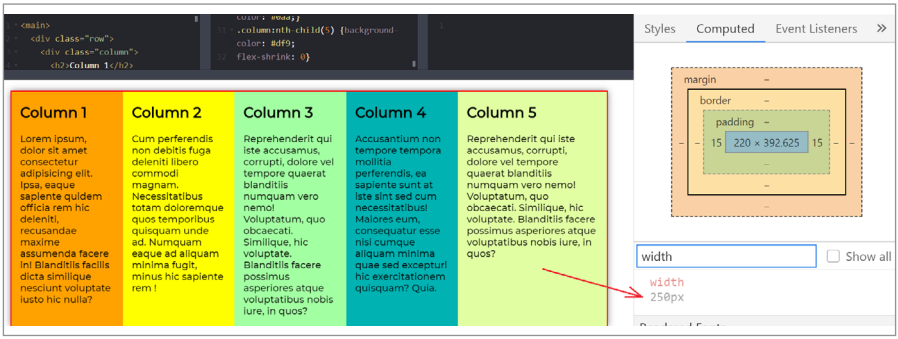


Рисунок 33

Ссылка на пример на codepen.io.

Если же у одного из элементов flex-shrink:2, то этот элемент станет уже, чем его соседи. Расчет сжимания будет несколько сложнее для восприятия, т.к. 250px: 6 = 41.66px, т.к. цифра 6 берется путем складывания коэффициентов flex-shrink для всех flex-элементов, т.е. 1+1+1+1+2 = 6.

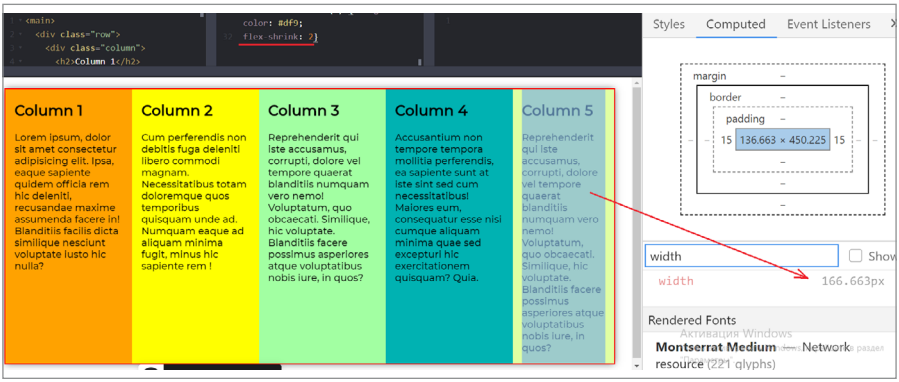


Рисунок 34

А затем от ширины каждого элемента браузер отнимет 41.66px: 250px – 41.66px = 208.34, а для элемента с flex-shrink:2 уменьшение будет больше, а ширина — меньше: 250px – 41.66*2 = 166.68px. Опять же — из-за размеров контента эта величина может изменяться (рис. 34).

Обобщенное свойство flex

Свойство flex позволяет задать сразу 3 предыдущих значения через пробел в такой последовательности:

```
flex: flex-grow flex-shrink flex-basis;
```

Значение по умолчанию: flex: 0 1 auto;

Варианты написания:

```
/* Одно значение, число без размерности: flex-grow */
flex: 2;
/* Одно значение, ширина/высота: flex-basis */
flex: 250px;
/* Два значения: flex-grow | flex-basis */
flex: 1 30px;
/* Два значения: flex-grow | flex-shrink */
flex: 2 2;
/* Три значения: flex-grow | flex-shrink | flex-basis */
flex: 2 1 25%;
```

Свойство order

Свойство order позволяет изменить последовательность расположения элементов. Это свойство определяет порядок следования flex-элементов и назначается в виде целочисленного значения как положительного, так и отрицательного. По умолчанию order имеет значение

0, и элементы размещаются в той последовательности, в которой они расположены в html-разметке. Если вы назначаете какому-либо элементу значение order:1, он переместится в конец, если order: -1 — в начало.

Свойство align-self

Свойство align-self меняет выравнивание отдельно взятого flex-блока по поперечной оси по сравнению со свойством flex-контейнера align-items. Варианты значений:

```
align-self: stretch | flex-start | flex-end | center |
            baseline
```

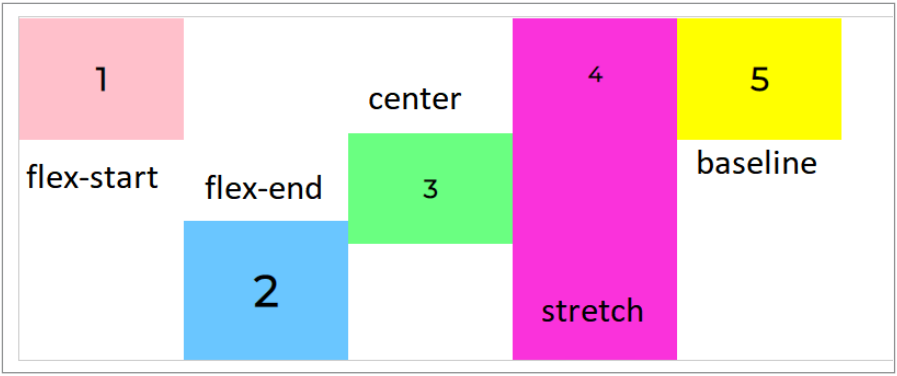


Рисунок 35

Полезные ссылки:

- CSS Flexible Box Layout Module — standart;
- A Complete Guide to Flexbox;
- The Difference Between Width and Flex Basis;
- Basic concepts of flexbox;
- Flex-grow is weird. Or is it?

Практические примеры использования Flexbox

Используем модель Flexbox для верстки посадочной страницы, состоящей из 5 блоков с информацией, шапки, меню и подвала.

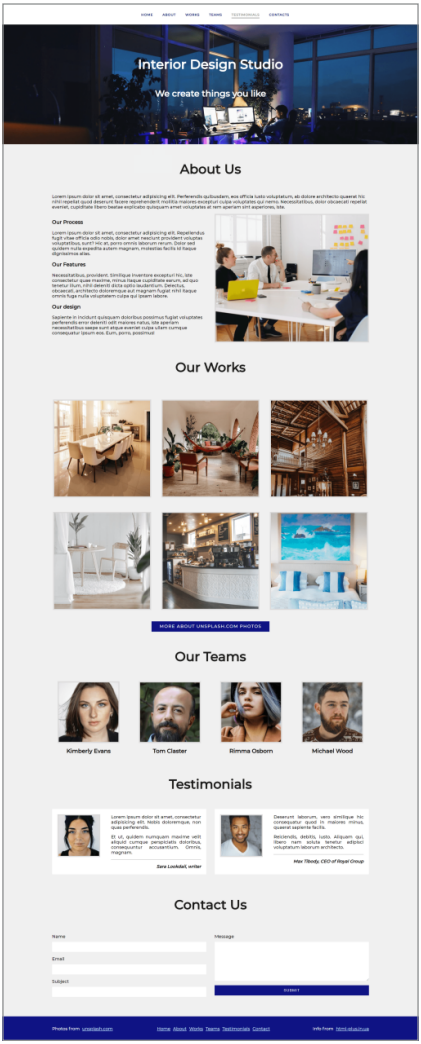


Рисунок 36

Для того чтобы вы могли посмотреть по отдельности на то, как работают стили для каждого из блоков, содержимое основного файла *index.html* будет разделено на несколько файлов (*gallery.html*, *testimonials.html*, *teams.html*, *contact.html*). Кроме того, для каждого из этих файлов будет создан отдельный css-файл (*gallery.css*, *testimonials.css*, *teams.css*, *contact.css*) в папке *css*. Все файлы вы найдете в папке *examples* этого урока. Они будут последовательно подключены в *index.html*.

Рабочий пример страницы вы найдете по ссылке на сайте codepen.io.

Такой подход не является стандартным при верстке страниц сайта. Обычно все стили для сайта содержатся в одном или 2-х css-файлах. В данном случае это сделано исключительно в целях обучения.

Общие стили для всех файлов запишем в файле *common.css* и подключим его во все html-файлы.

Рассмотрим внимательно наш макет. С точки зрения визуального распределения информации на странице можно выделить расположение текста и картинок в 2, 3 и 4 колонки. Поэтому мы будем использовать повторяющиеся классы для формирования колонок. Кроме того, нам понадобится класс для ограничения максимального пространства по ширине для больших экранов (*.container*). Мы также используем нестандартный шрифт Montserrat, импортировав его с помощью директивы @import из Google Fonts. Общие классы для форматирования:

```
@import url(https://fonts.googleapis.com/
                    css?family=Montserrat:500);
*,*::before,*::after { box-sizing: border-box;}
```

```
body {
    margin: 0;
    background-color: #f0f0f0;
    font-family: Montserrat, Arial, sans-serif;
    font-weight: 500;
}
.container {
    max-width: 1200px;
    margin: 0 auto;
    padding-left: 15px;
    padding-right: 15px;
}
```

Для формирования колонок нам понадобится 3 класса — один для flex-контейнера (.row), второй и третий (.column-2 и .column-3) — для колонок разной ширины, которые будут flex-элементами. Ширину каждой колонки определяем, исходя из 100% ширины родительского элемента путем деления на 2 или на 3 и задаем ее не с помощью свойства width, а с помощью свойства flex. Для колонок также зададим внутренние отступы в 15px слева и справа, а для flex-контейнера с классом .row — отрицательные внешние отступы в 15px, чтобы компенсировать padding-left и padding-right колонок.

```
.row {
    display: -webkit-flex;
    display: -ms-flex;
    display: flex;
    flex-wrap: wrap;
    margin-left: -15px;
    margin-right: -15px;
}
```

```
[class*="column-"] {
   padding-left: 15px;
   padding-right: 15px;
   min-width: 1px;
   max-width: 100%;
}
.column-2 { flex: 0 0 50%; }
.column-3 { flex: 0 0 33.33%; }
```

Также в общих классах мы опишем цвета ссылок, класс для оформления заголовков разделов (.heading), классы для отзывчивых изображений (.img-fluid) и изображений в рамке (.img-border), класс для центрирования текста (.text-center), для добавления внешних отступов снизу (.mb) и размещения flex-элементов по ширине flex-контейнера (.justify-content-between). Кроме того, нам понадобится класс для кнопки отправки формы и ссылки в виде кнопки (.btn). Стили описаны в файле *common.css* и приведены ниже:

```
a {color: #0f1384;}
a:hover {color: #151abc;}
.mb {margin-bottom: 20px;}
.text-center {text-align: center;}
.heading {
    font-size: 3rem;
    line-height: 1.5;
    text-align: center;
    margin: 3.5rem 0;
    color: #1a1a1a;
}
.justify-content-between { justify-content:
                           space-between; }
```

```
.img-fluid {
    max-width: 100%;
    height: auto;
}
.img-border {
    border: 4px solid #dbdbdb;
    transition: .5s;
}
.tab:hover .img-border,
.testimonial:hover .img-border,
.img-border:hover {
    border-color: #fff;
    box-shadow: 0 0 12px rgba(0, 0, 0, 0.38);
}
.btn {
    display: inline-block;
    height: 38px;
    padding: 0 30px;
    color: #fff;
    text-align: center;
    line-height: 38px;
    letter-spacing: .1rem;
    text-transform: uppercase;
    text-decoration: none;
    background: #0f1384;
    cursor: pointer;
    transition: .5s;
}
.btn:hover { background: #8488f8;
             text-decoration: none; }
```

Далее нам необходимо создать файл *index.html* и разместить в нем такую разметку:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <title>Interior Design Studio</title>
    <link rel="stylesheet" href="css/common.css">
</head>

<body>
    <header id="header">
        <h1>Interior Design Studio</h1>
        <h2>We create things you like</h2>
    </header>

    <section id="about">
        <div class="container">
           <h2 class="heading">About Us</h2>
           <p>Lorem ipsum dolor sit amet, consectetur
             adipisicing elit. Perferendis quibusdam,
             eos officia iusto voluptatum, ab dolore
             architecto quaerat hic nihil repellat
             quod deserunt facere reprehenderit
             mollitia maiores excepturi culpa
             voluptates qui nemo. Necessitatibus,
             dolor obcaecati repellat eveniet,
             cupiditate libero beatae explicabo
             quisquam amet voluptates at rem aperiam
             sint asperiores, iste.
           </p>
           <div class="row">
              <div class="column-2">
                 <h3>Our Process</h3>
           <p>Lorem ipsum dolor sit amet, consectetur
             adipisicing elit. Repellendus fugit vitae
             officia odio nobis, dolor amet nesciunt
             provident voluptas voluptatibus, sunt?
             Hic at, porro omnis laborum rerum.
             Dolor sed quidem nulla expedita autem
             magnam, molestias facilis id itaque
             dignissimos alias.
```

```
            </p>
                  <h3>Our Features</h3>
                  <p>inventore excepturi hic, iste
                     consectetur quae maxime, minus
                     itaque cupiditate earum, ad quo
                     tenetur illum, nihil deleniti
                     dicta optio laudantium.
                     Delectus, obcaecati, architecto
                     doloremque aut magnam fugiat
                     nihil itaque omnis fuga
                     nulla voluptatem culpa qui ipsam
                     labore.
                  </p>
                  <h3>Our design</h3>
                  <p>Sapiente in incidunt quisquam
                     doloribus possimus fugiat
                     voluptates perferendis
                     error deleniti odit maiores
                     natus, iste aperiam
                     necessitatibus saepe sunt atque
                     eveniet culpa ullam cumque
                     consequatur ipsum eos.
                     Eum, porro, possimus!
                  </p>
               </div>
               <div class="column-2">
                  <img src="https://source.unsplash.com/
                       MUZFKa_mttU/600x500"
                       class="img-fluid img-border"
                       alt="About us">
               </div>
            </div>
         </div>
      </section>
   </body>

</html>
```

В разметке мы будем вставлять фото с *unsplash.com*, но все изображения, использованные в примере, вы также можете найти в папке *examples/images*.

В результате мы получим страницу, на которой информация в <section id="about"> будет уже разбита на 2 колонки за счет использования такого кода:

```
.row { display: flex; }
.column-2 { flex: 0 0 50%; }
```

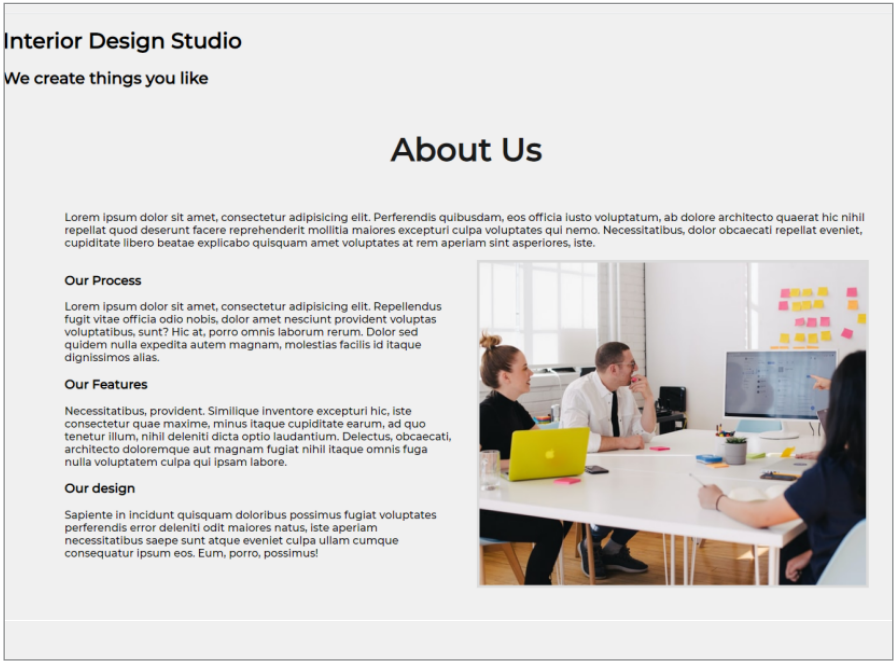


Рисунок 37

Аналогичным образом оформим блок Contacts, разделив поля формы между двумя колонками (вы найдете ее также в файле *contact.html*).

Разметка:

```
<section id="contacts">
<div class="container">
    <h2 class="heading">Contact Us</h2>
    <form name="contactForm">
       <div class="row">
          <div class="column-2">
              <p><label for="username">Name</label>
                 <input type="text" id="username"
                        name="username">
              </p>
              <p><label for="useremail">Email</label>
                 <input type="email" id="useremail"
                        name="useremail">
              </p>
              <p><label for="subject">Subject</label>
                 <input type="email" id="useremail"
                        name="subject">
              </p>
          </div>
          <div class="column-2">
             <p><label for="usermessage">
                   Message
                </label>
                <textarea class="u-full-width"
                          id="usermessage"
                          name="usermessage">
                </textarea>
             </p>
             <p><input class="btn" type="submit"
                       value="Submit">
             </p>
          </div>
        </div>
    </form>
</div>
</section>
```

Расположение в 2 колонки мы получили, кнопка также выглядит симпатично, но со всеми остальными полями нужно исправлять ситуацию.

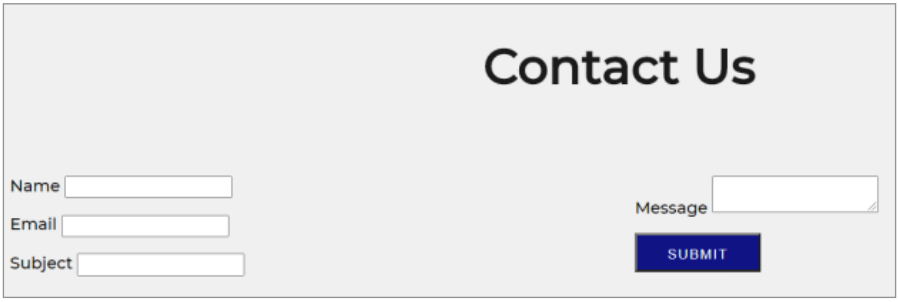


Рисунок 38

Вы можете найти ряд css-свойств в файле *contact.css* и подключить их к *index.html*:

```
<link rel="stylesheet" href="css/contact.css">

label {display: inline-block; margin-bottom: 10px;}

input, textarea {
    height: 38px;
    width: 100%;
    padding: 6px 10px;
    background-color: #fff;
    border: none;
    box-shadow: none;
    border-radius: 0;
    outline: none;
}

input:focus, textarea:focus {  box-shadow: 0 0 12px
      rgba(15, 19, 132, 0.38);  }
textarea { min-height: 145px; }
```

Внешний вид текстовых полей сразу изменился.

Рисунок 39

Форматирование шапки сайта

Далее необходимо поработать со стилями шапки сайта (элемент <header id="header">). Для того чтобы задавать эти стили, создадим в папке *css* файл *style.css* и подключим его в *index.html*:

```
<link rel="stylesheet" href="css/style.css">
```

В этом файле будем писать стили, относящиеся только к файлу *index.html*. Начнем с css-свойств для header. Для шапки сайта зададим темно-синий цвет фона и белый цвет текста, а также высоту в 500px и display: flex:

```
header {
    height: 500px;
    background-color: #0f1384;
    color: #fff;
    display: -webkit-flex;
    display: -ms-flex;
    display: flex;
}
```

Результат будет таким:

Рисунок 40

Это не совсем тот результат, который хотелось бы получить, т.к. 2 заголовка (h1 и h2), которые были в шапке, разместились рядом друг с другом за счет использования свойства display: flex. Мы же хотели разместить их друг под другом по центру header-а и по горизонтали, и по вертикали. Поэтому добавляем к уже написанным свойствам следующие:

```
header {... ;
    flex-direction: column;
    justify-content: center;
    -ms-align-items: center;
    align-items: center;
}
```

Результат теперь соответствует ожиданиям.

Рисунок 41

Осталось добавить фоновую картинку, заменив свойство background-color на background и добавить немного анимации к заголовкам. Итоговые стили:

```
@keyframes moveLeft {
    0% { opacity: 0; transform: translateX(-100%); }
}

@keyframes moveRight {
    0% { opacity: 0; transform: translateX(100%); }
}

header {
    height: 500px;
    background: url(https://source.unsplash.com/
                    NVHDSYqBZCE/1920x1080) no-repeat
                    center #0f1384;
    background-size: cover;
    color: #fff;
    display: -webkit-flex;
    display: -ms-flex;
    display: flex;
    flex-direction: column;
    justify-content: center;
    -ms-align-items: center;
    align-items: center;
}

h1 {
    font-size: 3em;
    animation: moveLeft .5s ease-out .3s backwards;
}

header h2 {
    font-size: 2em;
    animation: moveRight .5s ease-out .8s backwards;
}
```

Обратите внимание на то, что в стилях использованы вендорные префиксы -webkit- и -ms-. Они необходимы для поддержки старых версий браузеров Chrome, Safari и Internet Explorer 10-11. Далее в коде мы не будем их использовать, но вы должны понимать, что опуская эти префиксы, вы отказываетесь от нормального отображения свойств Flexbox в ряде браузеров, которыми продолжает пользоваться от 0,5 до 1% пользователей по всему миру.

Header теперь выглядит так:

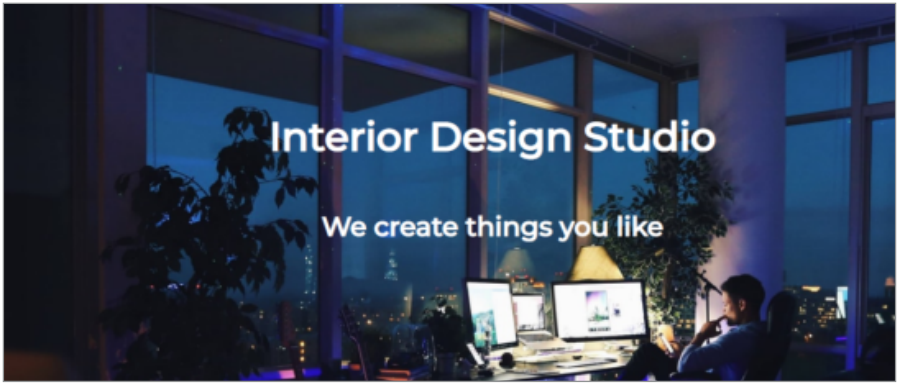


Рисунок 42

Раздел Our Works

В этом блоке у нас будет 2 ряда (.row) по 3 колонки в каждой. Сначала сделаем разметку:

```
<section id="portfolio">
    <div class="container">
      <h2 class="heading">Our Works</h2>
      <div class="row">
         <div class="portfolio-item column-3"
                    data-descr="Kitchen design">
```

```
    <a  target="_blank"
        href="https://source.unsplash.com/
              4a-xheeuZA0/800x800">
       <img src="https://source.unsplash.com/
                 4a-xheeuZA0/500x500"
                 class="img-fluid"
            alt="Kitchen design">
    </a>
</div>
<div  class="portfolio-item column-3"
      data-descr="Terrace design">
    <a target="_blank"
       href="https://source.unsplash.com/
             00fCk2lZn1c/700x900" >
      <img src="https://source.unsplash.com/
                00fCk2lZn1c/500x500"
          alt= "Terrace design"
          class="img-fluid" >
    </a>
</div>
<div  class="portfolio-item column-3"
      data-descr="Wooden house design">
    <a target="_blank"
       href="https://source.unsplash.com/
             7MAaRjiGH7I/800x800">
       <img src="https://source.unsplash.
                 com/7MAaRjiGH7I/500x500"
            alt="Wooden house design"
            class="img-fluid">
    </a>
</div>
<div class="portfolio-item column-3"
     data-descr="White home design">
    <a target="_blank"
       href="https://source.unsplash.com/
             X5BWooeO4Cw/700x800">
       <img src="https://source.unsplash.com/
```

```
                        X5BWooeO4Cw/500x500"
                    alt="White home design"
                    class="img-fluid">
            </a>
        </div>
        <div class="portfolio-item column-3"
             data-descr="Cafe design">
            <a href="https://source.unsplash.com/
                    AYdJM-HiSUM/1000x700"
             target="_blank">
            <img class="img-fluid"
                 src="https://source.unsplash.com/
                      AYdJM-HiSUM/500x500"
                 alt="Cafe design">
            </a>
        </div>
        <div  class="portfolio-item column-3"
              data-descr="Bedroom design">
            <a target="_blank"
               href="https://source.unsplash.com/
                     WzmzPT_sRzw/800x800" >
            <img src="https://source.unsplash.com/
                      WzmzPT_sRzw/500x500"
                 class="img-fluid"
                 alt="Bedroom design">
            </a>
         </div>
      </div>
      <p class="text-center">
          <a  class="btn"
              target="_blank"
              href="http://html-plus.in.ua/img-from-
                   unsplash-used-in-html-pages/">
            More about unsplash.com photos
           </a>
      </p>
    </div><!-- /.container -->
</section>
```

Посмотрим, что получилось у нас только после создания разметки с использованием класса column-3. У нас уже есть размещение в 3 колонки.

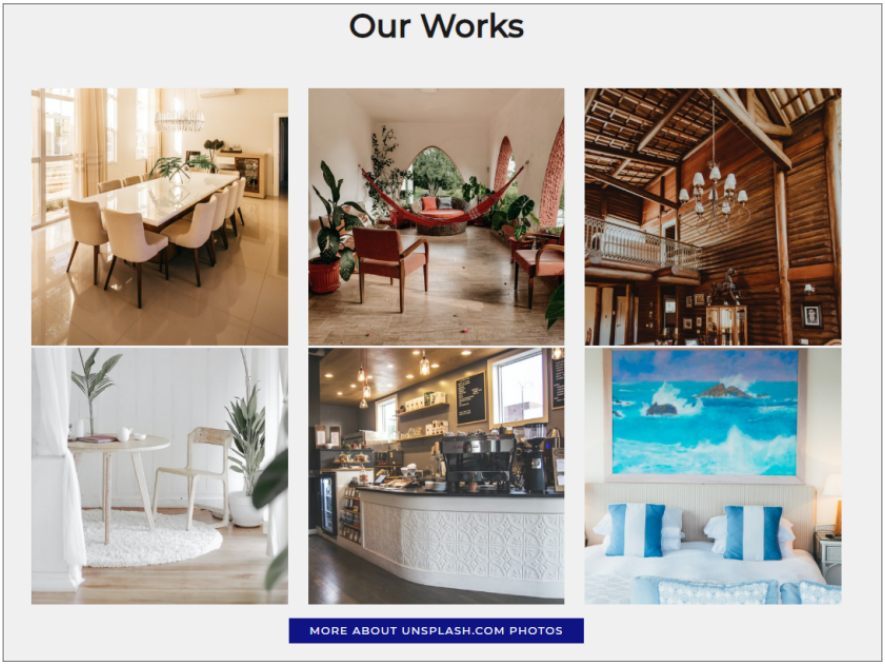


Рисунок 43

Остальные стили добавят отступы и позволят при наведении на каждую фотографию отобразить данные из атрибута data-descr.

```
.portfolio-item {
    margin: .5rem 0;
    padding-top: 15px;
    padding-bottom: 15px;
    position: relative;
    overflow: hidden;
```

```
    border: 4px solid transparent;
    transition: 400ms ease-out;
}
.portfolio-item img {
    display: block;
    border: 4px solid #dbdbdb;
    transition: 400ms ease-out;
}
.portfolio-item:hover {
    border-color: #fff;
    box-shadow: 0 0 24px rgba(0, 0, 0, 0.38);
}
.portfolio-item:hover img { transform: scale(1.15); }
.portfolio-item::after {
    content: attr(data-descr);
    display: block;
    position: absolute;
    left: 0;
    top:100%;
    width: 100%;
    padding: 20% 15px;
    background-color: rgba(255, 255, 255, 0.8);
    text-align: center;
    font-size: 1.5rem;
    transition: 500ms ease-out 200ms;
    pointer-events: none;
}
.portfolio-item:hover::after{  top: 20%;  }
```

Анимационные эффекты сложно передать с помощью скриншотов, но вы можете посмотреть на hover-эффекты в файле *gallery.html*. Подключить эти стили можно с помощью тега link:

```
<link rel="stylesheet" href="css/gallery.css">
```

Результат:

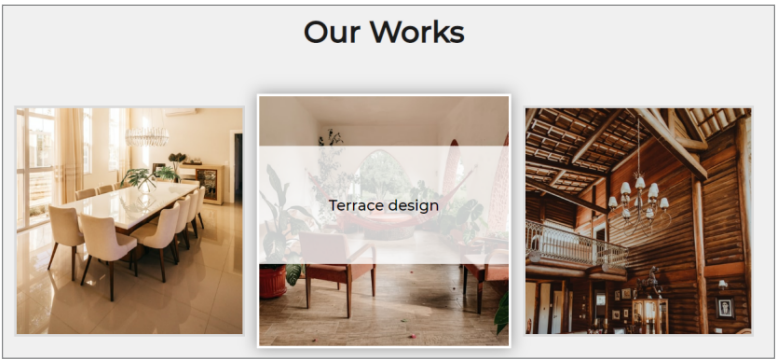


Рисунок 44

Раздел Teams

Блоки раздела Teams мы поместим внутрь div-а с классом .row, поэтому получим изначально такой внешний вид, в котором блоки будут сдвинуты к левому краю. Это размещение flex-элементов по умолчанию, если ширина (контент) каждого из них меньше, чем размер родительского элемента.

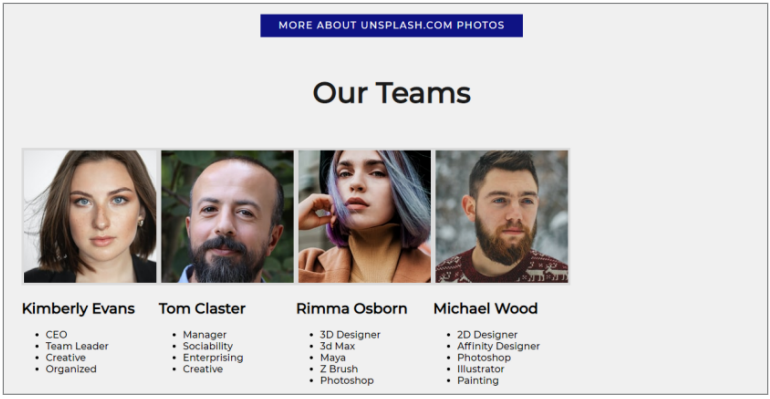


Рисунок 45

На скриншоте (рис. 45) видно, что блоки занимают меньше места, чем блоки портфолио вверху.

Разметка блока:

```
<section id="teams">
<h2 class="heading"> Our Teams</h2>
<div class="container">
    <div class="row">
        <div class="tab">
            <div class="tab-info">
                <img alt="Kimberly Evans"
                     class="img-border"
                     src="https://source.unsplash.
                          com/7ip-IOC2yvA/220x220" >
                <h2 class="name">Kimberly Evans</h2>
            </div>
            <ul class="tab-details">
                <li class="speciality">CEO</li>
                <li> Team Leader</li>
                <li> Creative</li>
                <li> Organized</li>
            </ul>
        </div>
        <div class="tab">
            <div class="tab-info">
                <img  alt="Tom Claster"
                      class="img-border"
                      src="https://source.unsplash.com/
                           eSjmZW97cH8/220x220">
                <h2 class="name">Tom Claster</h2>
            </div>
            <ul class="tab-details">
                <li class="speciality">Manager</li>
                <li> Sociability</li>
                <li> Enterprising</li>
                <li> Creative</li>
            </ul>
```

```
        </div>
        … <!-- еще 2 div class="tab" -->
    </div>
</div><!-- /.container -->
</section>
```

Изменим размер блоков с помощью свойства flex, задав в его составе flex-basis, flex-grow со значением 1 для блоков с классом .tab. Также оформим немного текстовые блоки с классами .name и .speciality.

```
.tab { flex: 1 0 20%; }
.name { font-size: 1.3em; }
.speciality { font-size: 1.1em; font-weight: bold;  }
```

Сейчас 4 блока у нас должны занимать 80% (20%*4) пространства flex-контейнера с классом .row, но за счет flex-grow: 1 они расширились до 25% (100%/4).

Это несколько улучшит ситуацию:

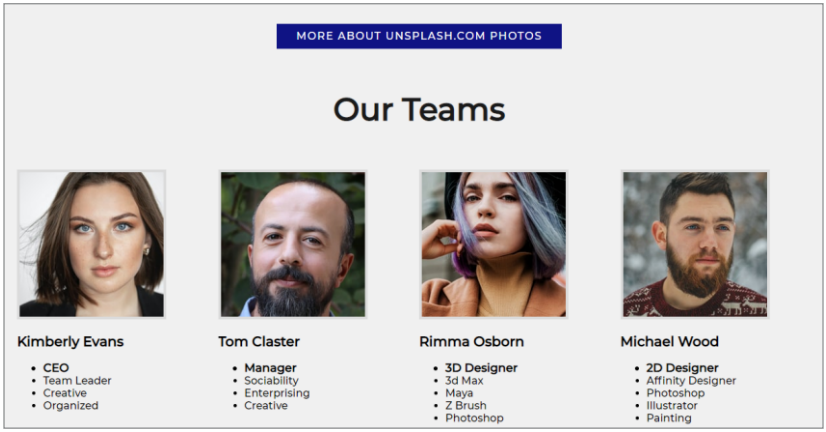


Рисунок 46

Целью наших манипуляций с css будет скрытие информации об участниках команды с помощью css-свойства opacity: 0 и ее отображение при наведении на div с классом .tab. Для начала нам нужно дописать ряд свойств для .tab и .tab-details.

```
.tab {
    flex: 1 0 20%;
    cursor: pointer;
    overflow: hidden;
    position: relative;
}
.tab-details {
/* opacity: 0; */
    padding: 20px;
    background-color: rgba(255, 255, 255, 0.57);
    list-style-type: none;
    line-height: 2;
    width: 200px;
    position: absolute;
    top: 0;
    left: 50%;
}
.tab-info { text-align: center; }
```

Результат применения стилей:

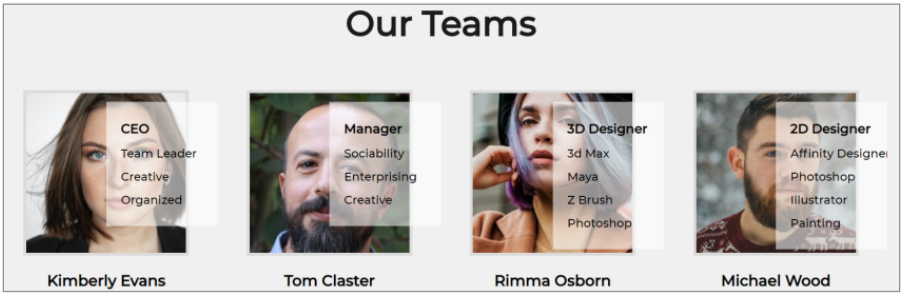


Рисунок 47

Теперь нам нужно раскомментировать opacity: 0 для .tab-details и добавить hover-эффекты с помощью css-свойства transition:

```
.tab {…;
transition: flex-basis 500ms ease-in-out; }
.tab:hover {
    flex-basis: 35%;
    overflow: visible;
}
.tab:hover .tab-details { opacity: 1; }
.tab-details {
    opacity: 0;
    …;
    transition: .5s; }
.tab:hover .tab-info {margin-left: 20%;}
.tab:hover .tab-details {left: 0;}
```

За счет свойства flex-grow: 1 и изменения flex-basis до 35% при наведении на .tab мы получаем интересные анимационные эффекты. Вы можете посмотреть на пример в файле *teams.html*. Для того чтобы стили применились в файле *index.html*, добавим их с помощью тега link:

```
<link rel="stylesheet" href="css/teams.css">
```

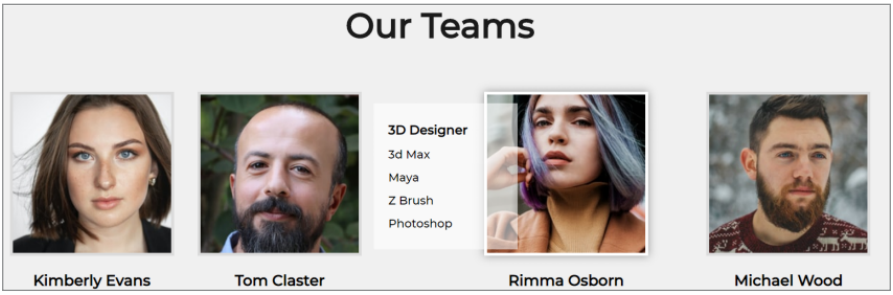


Рисунок 48

Раздел Testimonials

В этом разделе мы разместим 2 блока с отзывами внутри .

```
<section id="testimonials">
<div class="container">
    <h2 class="heading">Testimonials</h2>
    <div class="row">
       <div class="column-2 mb">
          <div class="testimonial">
             <img alt="Author 1"
                  class="testimonial-author
                         img-border"
                  src="https://source.unsplash.com/
                       EGVccebWodM/150x150">
             <div class="testimonial-body">
                <p> Lorem ipsum dolor sit amet,
                   consectetur adipisicing elit.
                   Nobis doloremque, non quas
                   perferendis.
                </p>
                <p> Et ut, quidem numquam maxime
                   velit aliquid cumque perspiciatis
                   doloribus, consequuntur
                   accusantium. Omnis, magnam.</p>
                <footer>
                   Sara Lookdail, writer
                </footer>
             </div>
          </div>
       </div>
       <div class="column-2 mb">
          <div class="testimonial">
             <img alt="Author 2"
                  class="testimonial-author
                         img-border"
                  src="https://source.unsplash.com/
                       iFgRcqHznqg/150x150">
```

```
            <div class="testimonial-body">
               <p> Deserunt laborum, vero similique
                  hic consequatur quod in maiores
                  minus, quaerat sapiente facilis.
               </p>
               <p> Reiciendis, debitis, iusto.
                  Aliquam qui, libero nam soluta
                  tenetur adipisci voluptatum
                  laborum architecto.
               </p>
               <footer>
                   Max Tibody, CEO of Royal Group
               </footer>
            </div>
         </div>
      </div>
   </div>
</div>
</section>
```

Внешне сейчас это выглядит так:

Рисунок 49

Используем для форматирования свойство display со значением inline-flex. Стили можно найти и подключить в файле *css/testimonials.css*.

```
.testimonial {
    display: inline-flex;
    transition: .5s;
    height: 100%;
    background-color: #fff;
}
.testimonial:hover {
    box-shadow: 0 0 24px rgba(0, 0, 0, 0.38);
}
.testimonial-author {
    display: block;
    margin: 20px;
}
.testimonial-body { margin: 20px; }
.testimonial-body p {
    margin-top: 0;
    text-align: justify;
}
.testimonial-body footer {
    text-align: right;
    font-weight: bold;
    font-style: italic;
    padding-top: 15px;
    border-top: 2px solid #dbdbdb;
}
```

После применения стилей фото выглядят растянутыми.

Рисунок 50

Мы уберем это с помощью свойства align-self: flex-start:

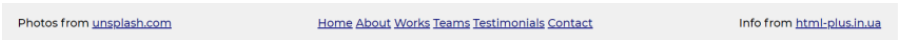
```
.testimonial-author { …;  align-self: flex-start; }
```

Теперь стало намного лучше:

Рисунок 51

Форматирование футера

Разметку футера вы сможете рассмотреть самостоятельно в файле *index.html*. Вместо колонок в нем использован , т.е. распределение пространства происходит за счет свойства justify-content: space-between.



Стили для footer мы добавляем в *style.css*:

```
#footer {
    background-color: #0f1384;
    color: #fff;
    margin-top: 3.5rem;
    padding: 20px 0;
}

#footer .row {   margin: 0; }
```

```
#footer a {
    color: #c0d9fc;
    display: inline-block;
    margin-left: 5px;
}
```

Результат применения css-стилей:

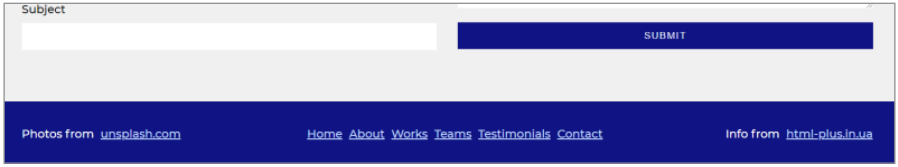


Рисунок 52

Форматирование меню

Чаще всего с этого пункта форматирование начинается. Однако в меню обычно используются ссылки на уже существующие в разметке блоки. Мы же эти блоки в виде разделов (<section>) с определенными id получили в процессе форматирования страницы. Поэтому мы создаем html-разметку в конце, но размещаем ее в самом верху *index.html* после открывающего тега <body>.

```
<div class="main-nav">
  <div class="container">
    <div class="mobile-toggle">
         <span></span>
         <span></span>
         <span></span>
    </div>
    <nav>
      <ul>
```

```
        <li><a class="menu-item" href="#header">
            Home</a>
        </li>
        <li><a class="menu-item" href="#about">
            About</a>
        </li>
        <li><a class="menu-item" href="#portfolio">
            Works</a>
        </li>
        <li><a class="menu-item" href="#teams">
            Teams</a>
        </li>
        <li><a class="menu-item" href= "#testimonials">
            Testimonials</a>
        </li>
        <li><a class="menu-item" href="#contacts">
            Contacts</a>
        </li>
      </ul>
    </nav>

  </div><!-- /.container -->
</div><!-- /.main-nav -->
```

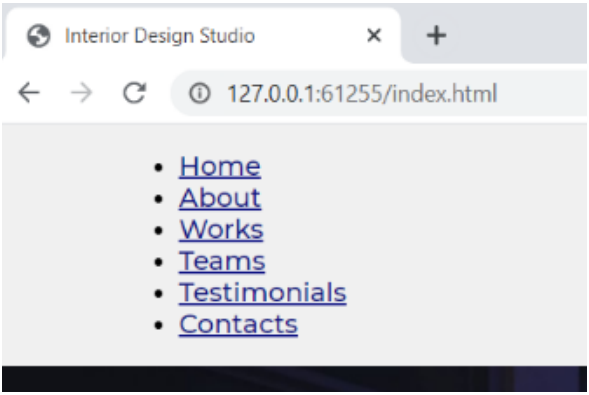


Рисунок 53

В css-стилях мы опять используем display: flex и justify-content: center, для того чтобы разместить пункты меню по центру. Плюс у нас будет иметь фиксированное позиционирование, чтобы к меню был доступ при прокрутке страницы.

```
.main-nav {
    position: fixed;
    top: 0px;
    z-index: 10;
    width: 100%;
    padding: 10px;
    background: #fff;
    box-shadow: 0 0 12px rgba(0, 0, 0, 0.38);
    min-height: 65px;
}

nav ul {
    list-style: none; padding: 0;
    display: flex;  justify-content: center;
}

nav ul li {
    margin-left: 35px;
    line-height: 1.5;
    letter-spacing: 1px;
}

nav ul a {
    text-transform: uppercase;
    font-size: 12px;
    font-weight: bold;
    text-decoration: none;
    border-bottom: 2px solid transparent;
    padding-bottom: 4px;
    transition: .5s;
}
```

```
nav ul a:hover {
    color: #989898;
    border-bottom-color: #989898;
}
.mobile-toggle {
    display: none;
    cursor: pointer;
    font-size: 20px;
    position: absolute;
    right: 22px;
    top: 20px;
    width: 30px;
}
```

Элемент необходим для отображения кнопки для меню для смартфонов. Кроме того, для смартфонов и планшетов нам понадобится переписать ряд стилей, которые изменят внешний вид нашей html-страницы в зависимости от ширины экранов.

Изменение стилей для смартфонов и планшетов

Во-первых, в блоке <head> в файле *index.html* необходимо добавить тег meta, запрещающий масштабирование страницы для смартфонов и планшетов и подключить еще один css-файл с медиа-запросами:

```
<meta name="viewport"
      content="width=device-width,
      user-scalable=no,
      initial-scale=1.0,
      maximum-scale=1.0,
      minimum-scale=1.0">
<link rel="stylesheet" href="css/responsive.css">
```

Screen Resolution Statistics

Screen resolution display size statistics. Most common screen resolutions:

Screen resolution	Display ratio	Usage	Screen size / type
1366x768	16:9	19.1%	14" Notebook / 15.6" Laptop / 18.5" monitor
1920x1080	16:9	9.4%	21.5" monitor / 23" monitor / 1080p TV
1280x800	8:5	8.5%	14" Notebook
320x568	9:16	6.4%	4" iPhone 5
1440x900	8:5	5.7%	19" monitor
1280x1024	5:4	5.5%	19" monitor
320x480	2:3	5.2%	3.5" iPhone
1600x900	16:9	4.6%	20" monitor
768x1024	3:4	4.5%	9.7" iPad
1024x768	4:3	3.9%	15" monitor
1680x1050	8:5	2.8%	22" monitor
360x640	9:16	2.3%	
1920x1200	8:5	1.7%	24" monitor
720x1280	9:16	1.6%	4.8" Galaxy S
480x800	3:5	1.1%	
1360x768	16:9	0.9%	
1280x720	16:9	0.9%	720p TV

Рисунок 54

В *responsive.css* мы должны указать определенные брейкпойнты для перестраивания внешнего вида нашего макета. Поскольку у нас одна страница, правил будет относительно немного. Подбирать мы их будем, исходя из статистики по разрешениям экрана. На скриншоте представлена статистика с сайта rapidtables.com, которая позволяет получить представление об устройствах,

на которых данные разрешения используются. Мы задали максимальную ширину контейнера не более 1200px, то есть на всех мониторах наша страница будет выглядеть в соответствии с теми стилями, что уже написаны. Теперь нужно перестроить внешний вид для планшетов (768 — 1024px) и смартфонов (экраны 480px и менее). В связи с большой популярностью iPhone, необходимо будет проверять еще такие разрешения, как 375 и 320px. Кроме того, нам нужно будет посмотреть, как расположение наших элементов выглядит при динамическом изменении размеров экрана, которое мы можем получить с помощью Инструментов разработчика в браузере (F12 или Ctrl +Shift +I).

Для планшетов выберем брейкпойнт в 960px. Такое разрешение имеет не слишком большое количество устройств, но именно до этого размера экрана хорошо выглядит блок Our Teams.

В media-запросе в первую очередь нам необходимо изменить высоту header и размеры загружаемого в него изображения, увеличить размер .column-3, убрать ряд правил для блока Our Works и перестроить отображение блоков .tab и .tab-details в Our Teams. Также несколько уменьшим фото и отступы в Testimonials.

```
@media (max-width: 960px) {
    header {height: 450px;
            background-image: url(https://source.
                unsplash.com/NVHDSYqBZCE/1000x450);
    }
   .column-3 {flex-basis: 50%; }
   .portfolio-item {margin-bottom: 0; padding-top: 0;}
```

```
    .portfolio-item:hover {
        box-shadow: none;
    }
    .portfolio-item:hover img {
       transform: scale(1.05);
    }
    .portfolio-item::after,
    .portfolio-item:hover::after {
        top: auto;
        width: calc(100% – 38px);
        left: 19px;
        bottom: 25px;
        padding: 20px 15px;
    }
    .testimonial-body, .testimonial-author {
        margin: 15px;
    }
    .testimonial-author {
        width: 120px;
    }
    .tab, .tab:hover {
        flex: 0 0 calc(50% – 30px);
     }
    .tab {
        background-color: #fff;
        padding-top: 20px;
        margin: 0 15px 25px;
    }
    .tab:hover .tab-info { margin-left: 0; }
    .tab-details {
        opacity: 1;
        position: static;
        background-color: transparent;
        width: auto;
        text-align: center;
        padding-top: 0;
    }
}
```

Вы можете копировать код построчно, чтобы видеть, как постепенно меняется внешний вид разных блоков (рис. 55–57).

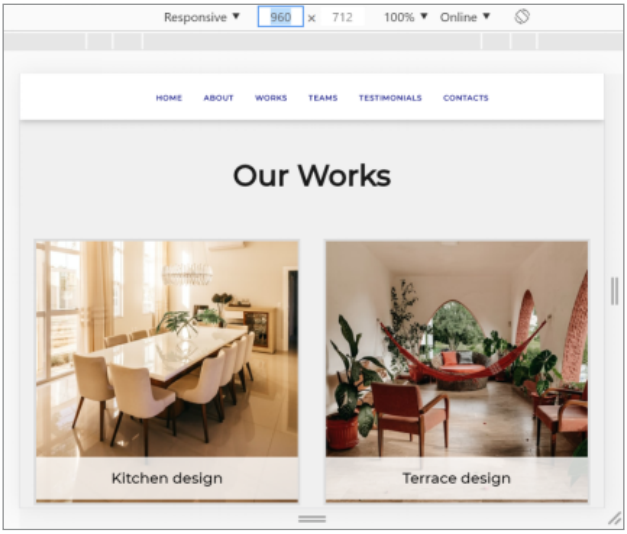


Рисунок 55

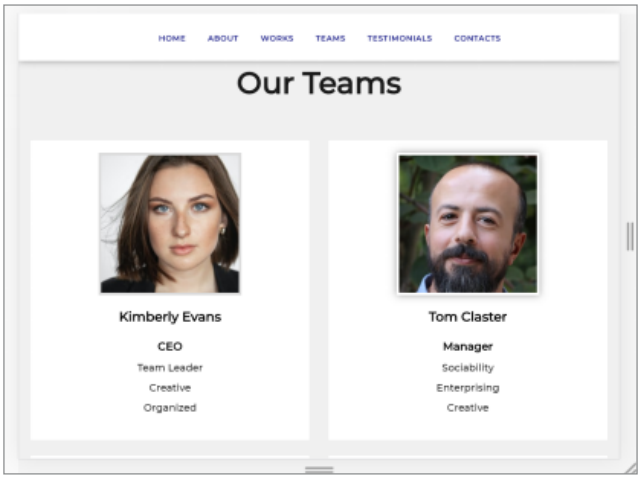


Рисунок 56

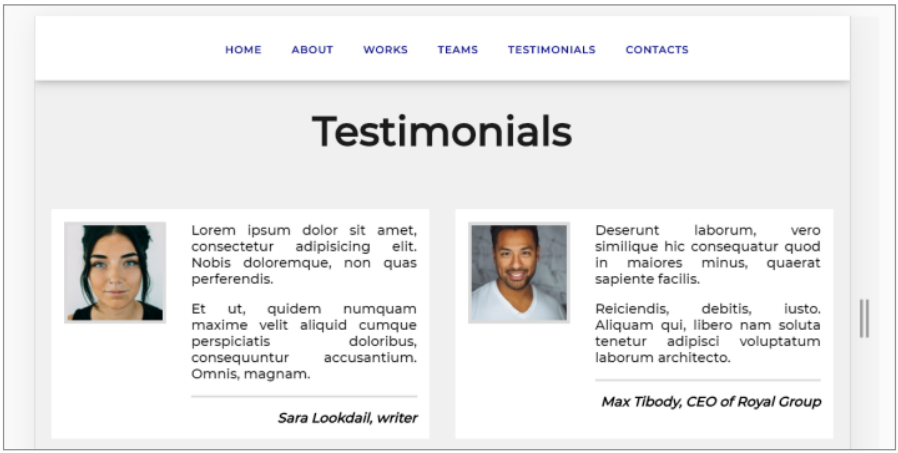


Рисунок 57

На экранах с разрешением от 768px мы поменяем отображение меню. Вместо 6 пунктов мы разместим кнопку-гамбургер с 3-мя полосками, которые образуют span-элементы внутри и напишем для них стили. Кроме того, мы уберем отрицательные отступы у .row, .column-2 у нас расширится до размеров родительского элемента за счет flex-basis: 100%. В футере также вложенные div-ы займут все доступное пространство внутри родителя с выравниванием текста по центру.

```
@media (max-width: 768px) {
    header {padding: 15px; text-align: center; }
    .row { margin-left: 0; margin-right: 0; }
    .column-2 { flex-basis: 100%; }
    .mobile-toggle { display: block; }
    .mobile-toggle span {
        width: 30px;
        height: 4px;
```

```
        margin-bottom: 6px;
        background: #000000;
        display: block;
    }
    nav {display: none; }
    .main-nav.open nav {display: block;  }
    nav ul { display: block;  }
    nav ul li {
        width: 100%;
        padding: 7px 0;
        margin: 0;
        text-align: center;
    }
    nav ul li:first-child {  margin-top: 30px;  }
    #footer div { width: 100%; text-align: center;  }
}
```

Внешний вид страницы снова изменится (рис. 58–60).

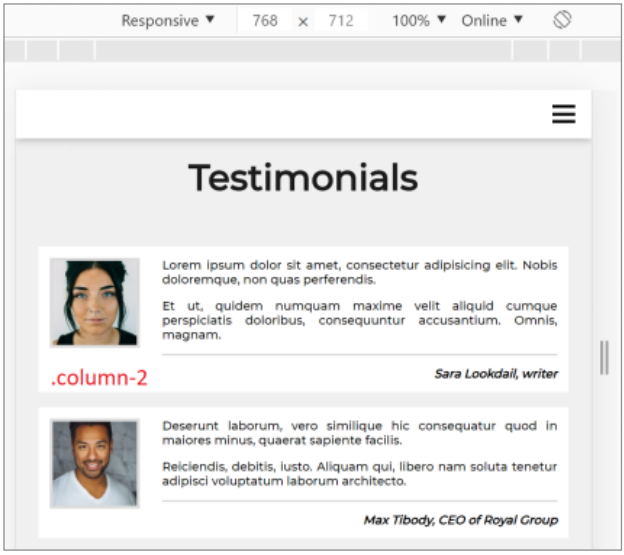


Рисунок 58

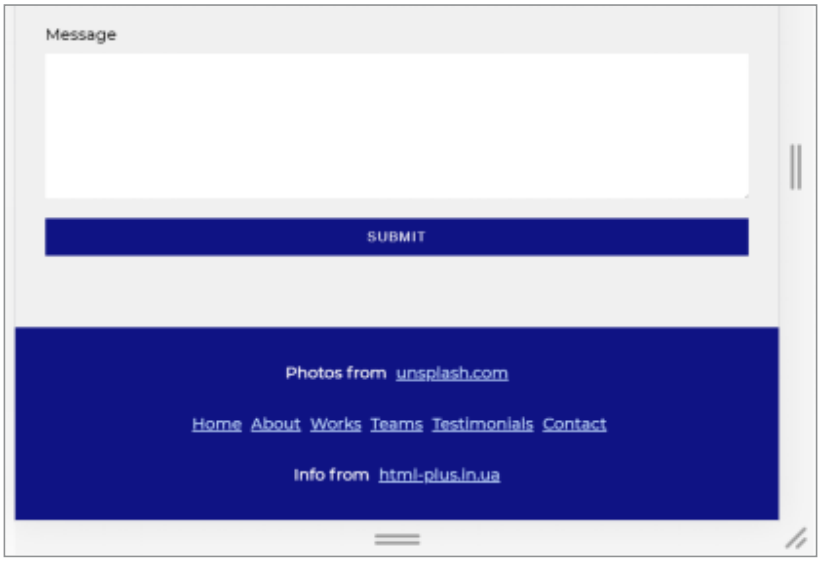


Рисунок 59

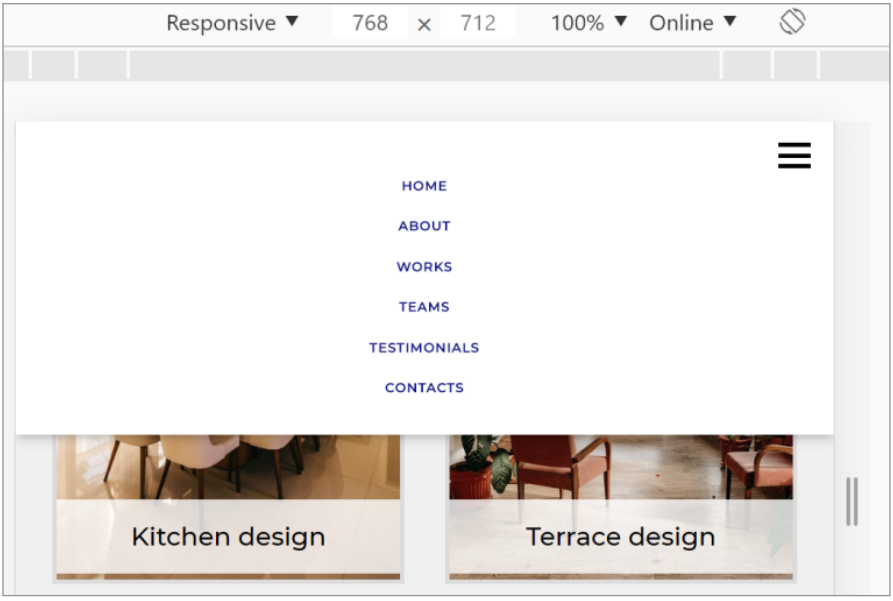


Рисунок 60

При клике на кнопку-гамбургер наш меню разворачивается за счет кода на JavaScript. Для этого нам понадобится подключить небольшой скриптовый файл перед закрывающим тегом </body>:

```
<script src="js/main.js"></script>
```

Этот скрипт добавляет/удаляет класс .open у блока с классом .mobile-toggle, а css-код делает все остальное.

Код для смартфонов мы запишем для разрешния от 480px:

```
@media (max-width: 480px) {
    h1 {  font-size: 2em; }
    header h2 { font-size: 1.5em; }
    .heading { font-size: 2.5rem; margin: 2rem 0;}
    header { height: 350px;
             background-image: url(https://source.
                               unsplash.com/NVHDSYqBZCE/
                               500x350); }
    [class*="column-"] { flex-basis: 100%; }
    .tab, .tab:hover { flex-basis: calc(100% – 30px); }
    .testimonial {
          display: block;
          padding-top: 5px;
          padding-bottom: 15px;
    }
    .testimonial-author { margin: 20px auto; }

}
```

Для небольших экранов мы меняем размеры заголовков, высоту header и размеры загружаемого в него фонового изображения, увеличиваем размеры блоков до 100% и изменяем внешние и внутренние отступы.

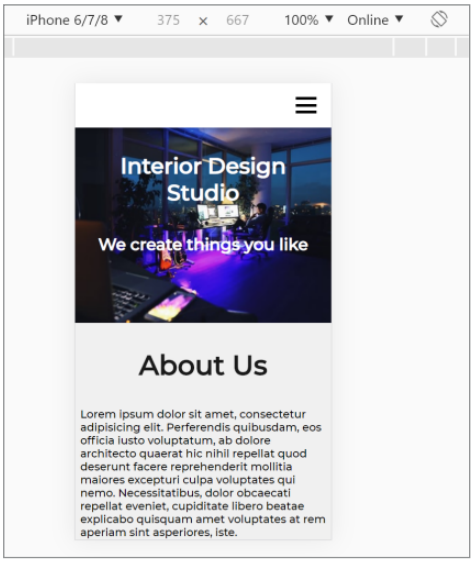


Рисунок 61

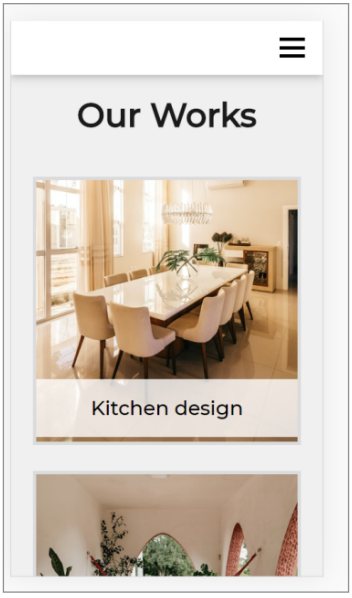


Рисунок 62

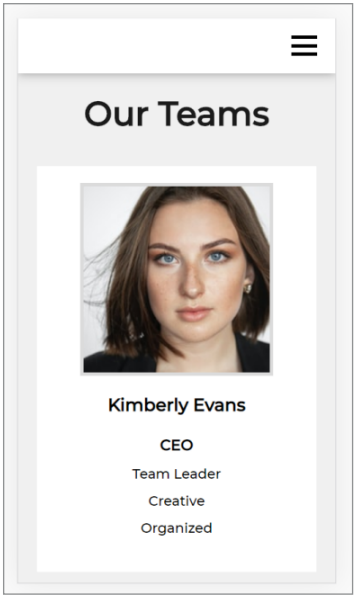


Рисунок 63

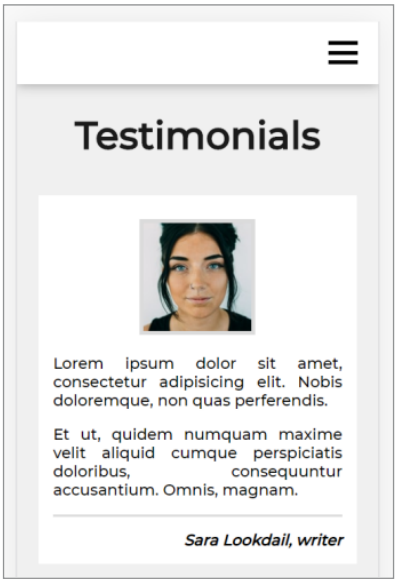


Рисунок 64

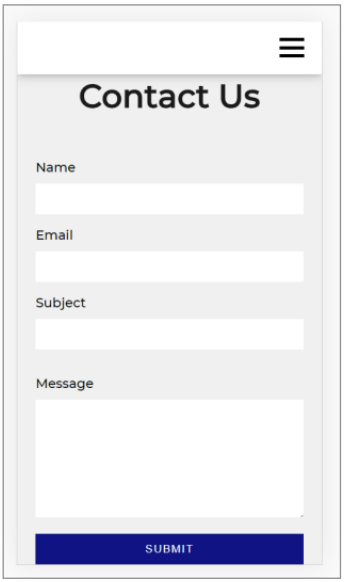


Рисунок 65

Полезные ссылки:

- Статистика разрешений экрана по годам;
- Глобальная статистика разрешений экрана за год;
- Разрешения экранов с указанием устройств;
- https://itchief.ru/lessons/html-and-css/css-media-queries.

Домашнее задание

Дома вам необходимо сверстать страницу, представляющую собой мини-портфолио фотографа. Внешний вид на стационарных компьютерах представлен ниже. Скриншоты для планшетов и телефонов вы найдете в папке *HW*.

Фотографии можно взять с unsplash.com или из папки *images*. В ней вы также найдете логотип и иконку, появляющуюся при наведении на нижний ряд фото.

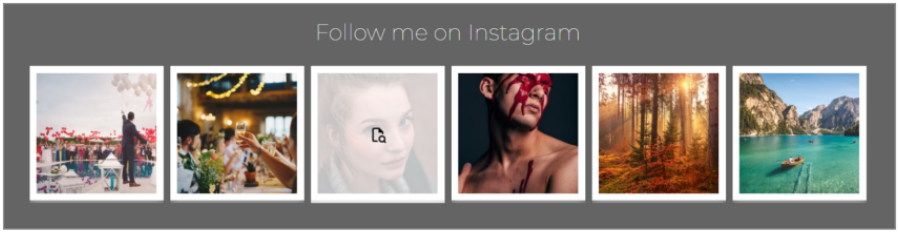


Рисунок 62

При наведении на фото из раздела My Last Works должен появляться текст на полупрозрачном белом фоне. Текст вы найдете в файле *homework_text.txt* в папке *HW*.

Рисунок 63

Постарайтесь использовать одни и те же классы для форматирования однотипных объектов. Имеется в виду то, что в макете есть размещение элементов в 2 и 3 колонки в header, About Me, My Last Works.

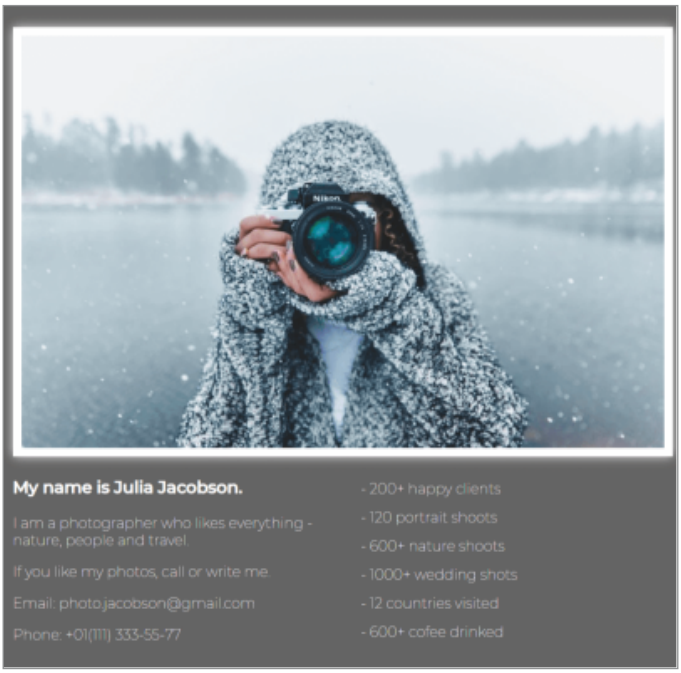


Рисунок 64

Обратите внимание, что при изменении разрешения на планшетах, в блоке About Me колонки с текстом перестраиваются под изображение. Подумайте, как разместить блоки в этом разделе.

Для меню вы можете использовать разметку, подобную приведенной в уроке + js-файл. Если вы не сможете показать меню при клике, оставьте просто иконку-гамбургер.


Julia Jacobson
The Photographer
About
Works
Instagramm
About Me
My name is Julia Jacobson.
I am a photographer who likes everything - nature, people and travel
If you like my photos, call or write me.
Email: photo.jacobson@gmail.com
Phone: +01(11) 333-55-77
- 200+ happy clients
- 120 portrait shoots
- 600+ nature shoots
- 1000+ wedding shots
- 12 countries visited
- 600+ cofee drinked
My Last Works
Morning coffee
Cupiditate nesciunt suscipit sunt molestiae pariatur nihil explicabo voluptas assumenda...
Landscape Photo
Architecto debitis fugit quos animi possimus perspiciatis vel aperiam...
Follow me on Instagram
© Copyright 2018-2020 Julia Jacobson

Рисунок 65

Урок 8.
Модель Flexbox