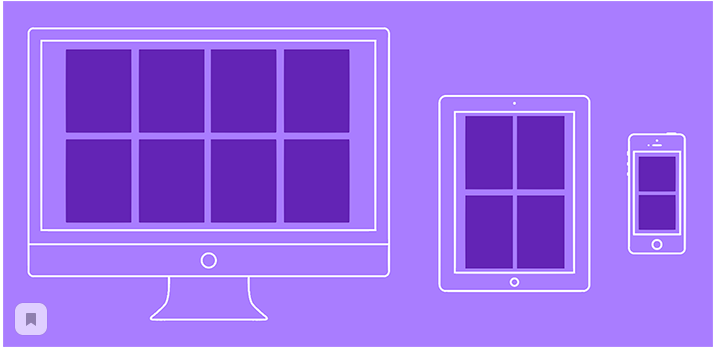
**3. Создание карт с flexbox**

Карты универсальны, визуально привлекательны и легко взаимодействуют как на больших, так и на малых устройствах, что идеально подходит для адаптивного дизайна. Каждая карта действует как контейнер содержимого, который легко масштабируется вверх или вниз. Поскольку размеры экрана становятся меньше, то они обычно уменьшают количество карт в строке, и они начинают складываться вертикально. Дополнительная гибкость карт возрастает по мере того как они могут быть фиксированной или переменной высотой.

**Как создать макет**

Мы создадим макет карты Flexbox, который имеет ряд из четырех горизонтальных контейнеров на больших экранах, два на средних и один столбец для небольших устройств.



Ниже приведен код для создания базового макета для отображения четырех карт.  В него не включено внутреннее содержимое карты, поэтому не забудьте поместить туда некоторое начальное содержимое (разное для разных карт). Карт можно добавить больше, если вы хотите видеть поведение с несколькими строками карт.

Раздел с классом .cards- это то, на что мы будем ориентироваться в первую очередь. Свойство отображения контейнера - это то, что нам нужно изменить flex.

**Вот HTML, с которого начинаем:**

<header class="masthead clear">

<div class="centered">

<div class="site-branding">

<h1 class="site-title">Flexbox Demo: Card Layouts</h1>

<p>Text from <a title="Office Ipsum" href="http://officeipsum.com/index.php">Office Ipsum</a> </p>

</div><!-- .site-title -->

</div><!-- .centered -->

</header><!-- .masthead -->

<main class="main-area">

<div class="centered">

<section class="cards">

<article class="card">

<a href="#">

<picture class="thumbnail">

<img src="image-example-01.jpg" alt="A banana that looks like a bird">

</picture>

<div class="card-content">

<h2>Vacation Image 01</h2>

<p>...</p>

</div><!-- .card-content -->

</a>

</article><!-- .card -->

<article class="card">

<a href="#">

<picture class="thumbnail">

<img src="image-example-02.jpg" alt="Norwegian boller">

</picture>

<div class="card-content">

<h2>Vacation Image 02</h2>

<p>… </p>

<p>… </p>

</div><!-- .card-content -->

</a>

</article><!-- .card -->

<article class="card">

<a href="#">

<picture class="thumbnail">

<img src="example-03.jpg" alt="A dinosaur wearing an aluminium jacket">

</picture>

<div class="card-content">

<h2>Vacation Image 03</h2>

<p>… </p>

</div><!-- .card-content -->

</a>

</article><!-- .card -->

<article class="card">

<a href="#">

<picture class="thumbnail">

<img src="Image-example-04.jpg" alt="Chocolate filled boller">

</picture>

<div class="card-content">

<h2>Vacation Image 04</h2>

<p>….</p>

</div><!-- .card-content -->

</a>

</article><!-- .card -->

</section><!-- .cards -->

</div><!-- .centered -->

</main>

CSS, чтобы начать:

/\* Import the google web fonts you want to use \*/

<!--@import url('https://fonts.googleapis.com/css?family=Comfortaa|Patua+One');-->

body {

background: hsl(0, 0%, 90%);

}

body,

button,

input,

select,

textarea {

color: #404040;

font-family: 'Comfortaa', Arial, sans-serif;

font-size: 18px;

line-height: 1.5;

}

h1,

h2,

h3,

h4,

h5,

h6 {

clear: both;

font-family: 'Patua One', serif;

letter-spacing: 2px;

}

h2 {

font-size: 1rem;

color: #007489;

}

p {

margin-bottom: 1.5em;

color: #898989;

}

b,

strong {

font-weight: bold;

}

dfn,

cite,

em,

i {

font-style: italic;

}

blockquote {

margin: 0 1.5em;

}

address {

margin: 0 0 1.5em;

}

pre {

background: #eee;

font-family: "Courier 10 Pitch", Courier, monospace;

font-size: 15px;

font-size: 1.5rem;

line-height: 1.6;

margin-bottom: 1.6em;

max-width: 100%;

overflow: auto;

padding: 1.6em;

}

code,

kbd,

tt,

var {

font: 15px Monaco, Consolas, "Andale Mono", "DejaVu Sans Mono", monospace;

}

abbr,

acronym {

border-bottom: 1px dotted #666;

cursor: help;

}

mark,

ins {

background: #fff9c0;

text-decoration: none;

}

sup,

sub {

font-size: 75%;

height: 0;

line-height: 0;

position: relative;

vertical-align: baseline;

}

sup {

bottom: 1ex;

}

sub {

top: .5ex;

}

small {

font-size: 75%;

}

big {

font-size: 125%;

}

.masthead {

background-color: #005b75;

}

.masthead p, .masthead p a {

color: white;

}

.site-title {

color: white;

}

img {

display: block;

border: 0;

width: 100%;

height: auto;

}

.card {

background: white;

margin-bottom: 2em;

}

.card a {

color: black;

text-decoration: none;

}

.card a:hover {

box-shadow: 3px 3px 8px hsl(0, 0%, 80%);

}

.card-content {

padding: 1.4em;

}

.card-content h2 {

margin-top: 0;

margin-bottom: .5em;

font-weight: bold;

}

.card-content p {

font-size: 80%;

}

/\* Flexbox \*/

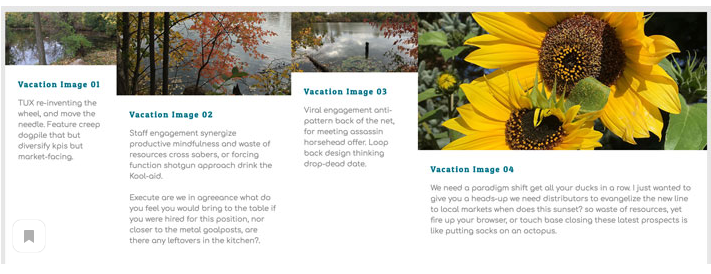
.cards {

display: flex;

justify-content: space-between;

}

Подключите файл стилей. Получим



почему каждая карточка имеет т разную ширину? Flexbox пытается выяснить, какая наименьшая ширина по умолчанию для каждой из карт. И из-за различных длин слов и других элементов дизайна, вы в конечном итоге имеете дело с блоками различной ширины и высоты.

Измените в конце файл CSS

.cards {

   display: flex;

   flex-wrap: wrap;

   justify-content: space-between;

}

По умолчанию все элементы flex будут пытаться поместиться в одну строку. Добавление flex-wrap: wrap;делает перенос элементов , потому что по умолчанию используется полная ширина.

Полная ширина отлично подходит для небольших устройств, так что давайте иметь это в виду, Когда мы меняем ширину, карты начинают выглядеть более ровными.

Теперь нам нужно добавить .card класс, чтобы стилизовать наши карты. Это можно сделать после стиля для .cards.

.cards {

   display: flex;

   flex-wrap: wrap;

   justify-content: space-between;

}

.card {

    flex: 0 1 24%;

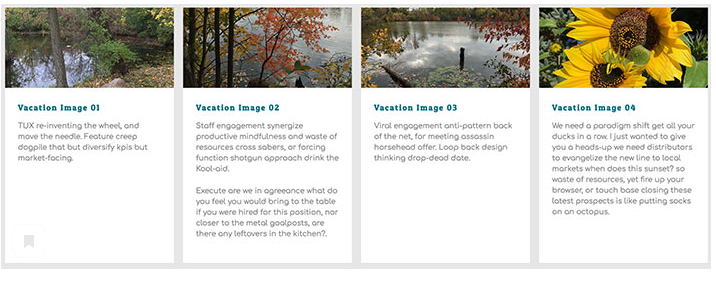
}

lex-growСвойство элемента flex указывает, какой объем пространства внутри контейнера flex должен занимать элемент.

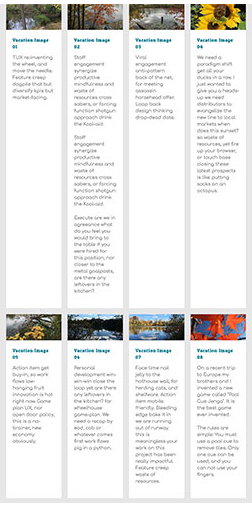
flex-shrinkСвойство указывает, как элемент будет сжиматься относительно остальных гибких элементов внутри того же контейнера.

flex-basisСвойство указывает начальный основной размер элемента flex.

свойство flex является стенографическим: flex-grow равно 0, flex-shrink равно 1 и widthравно 24%. Добавляя заданную ширину, это дает нам ряд из четырех карт с некоторым промежутком между ними.



Мы устанавливаем justify-contentсвойство для интервалов. Первый элемент отображает жесткий левый, второй и третий элементы отображаются в середине, а четвертый элемент отображает жесткий правый. Поскольку ширина карты составляет 24%, есть некоторое свободное место, так как наши четыре столбца на 24% не составляют 100%. У нас осталось 4%, чтобы быть точным. Эти 4% помещены поровну между каждым из деталей. Таким образом, у нас есть примерно 1,33% пространства между картами.



Уточним размер с помощью calc. Изменение flex-basisзначения для использования calc будет выглядеть примерно так:

.card {

    flex: 0 1 calc(25% - 1em);

}

браузер будет захватывать 25% пространства и удалять из него 1em, что делает карты немного меньше.

### Карты для небольших устройств

В настоящее время у нас есть четыре колонки на всех экранах, что на самом деле не лучшая практика. Если вы сделаете окно вашего браузера меньше, вы увидите, что четыре карты просто становятся более сплющенными на меньших экранах, что не идеально подходит для удобства чтения. К счастью, с media запросами все начнет выглядеть намного лучше.

**Добавьте в стили:**

@media screen and (min-width: 40em) {

    .cards {

   }

    .card {

    }

}

@media screen and (min-width: 60em) {

    .cards {

   }

    .card {

     }

}

@media screen and (min-width: 52em) {

    .centered {

    }

}

Уточним для min-width: 40em

@media screen and (min-width: 40em) {

    .cards {

        display: flex;

        flex-wrap: wrap;

        justify-content: space-between;

    }

    .card {

        flex: 0 1 calc(25% - 1em);

    }

}

С этими изменениями карты будут отображаться на полноэкранной ширине и укладываться друг под другом на любом экране меньше, чем около 640 пикселей в ширину. Если вы развернете окно браузера на что-либо выше этого, столбец из четырех возвращается. Это имеет смысл, потому что есть min-width40em, и именно здесь мы создали ряд из четырех карт.

Чего здесь не хватает, так это середины. Для среднего диапазона наличие двух карт подряд более читаемо, а не четыре сдавленные карты. Прежде чем мы определим ряд из двух карт, необходимо добавить еще один медиа-запрос, чтобы разместить самые большие экраны, которые будут иметь ряд из четырех карт.

|  |
| --- |
| @media screen and (min-width: 60em) {      .card {          flex: 0 1 calc(25% - 1em);      }  } |

Новый медиа-запрос с A min-width60em - это то, где будут объявлены четыре карты. The min-widthof 40em - это место, где будет объявлен ряд из двух карт. Магия происходит со значением flex calc50% - 1em.

|  |
| --- |
| @media screen and (min-width: 40em) {      .cards {          display: flex;          flex-wrap: wrap;          justify-content: space-between;      }        .card {          flex: 0 1 calc(50% - 1em);      }  } |

С этим простым изменением, карты теперь работают! Сожмите и разверните окно браузера, чтобы убедиться, что все выглядит правильно.

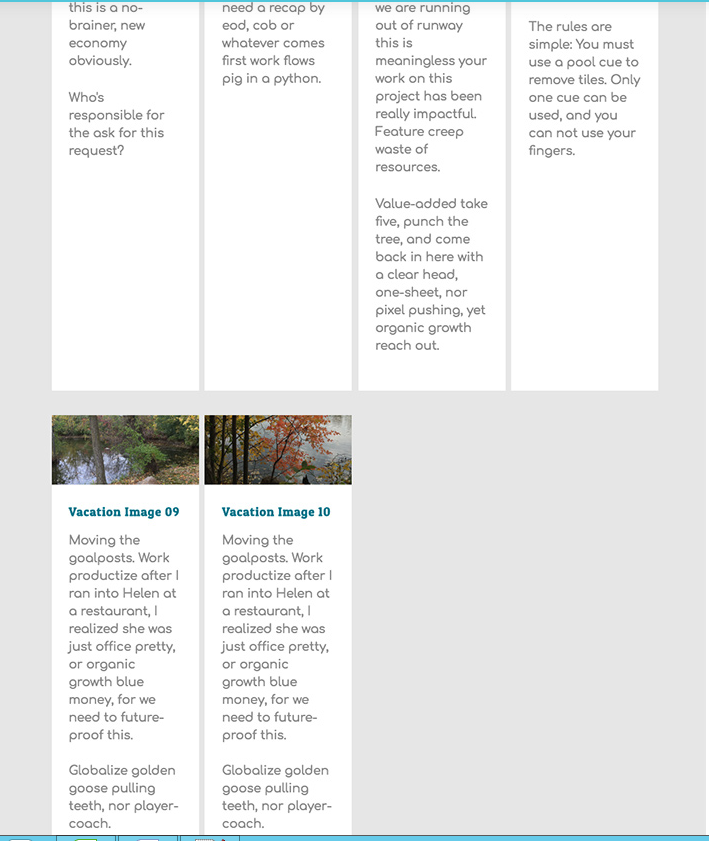
## Динамическое содержимое и последняя строка интервала карт

Иногда ваш контент планируется заранее, но если он является динамическим, то последний ряд карт может вести себя не так, как вы предполагали.  выше.



**Запросы мультимедиа-это то, где определяется количество карт:**

|  |
| --- |
| @media screen and (min-width: 40em) {      .card {         max-width: calc(50% -  1em);      }  }    @media screen and (min-width: 60em) {      .card {          max-width: calc(25% - 1em);      }  } |



**Итог**

