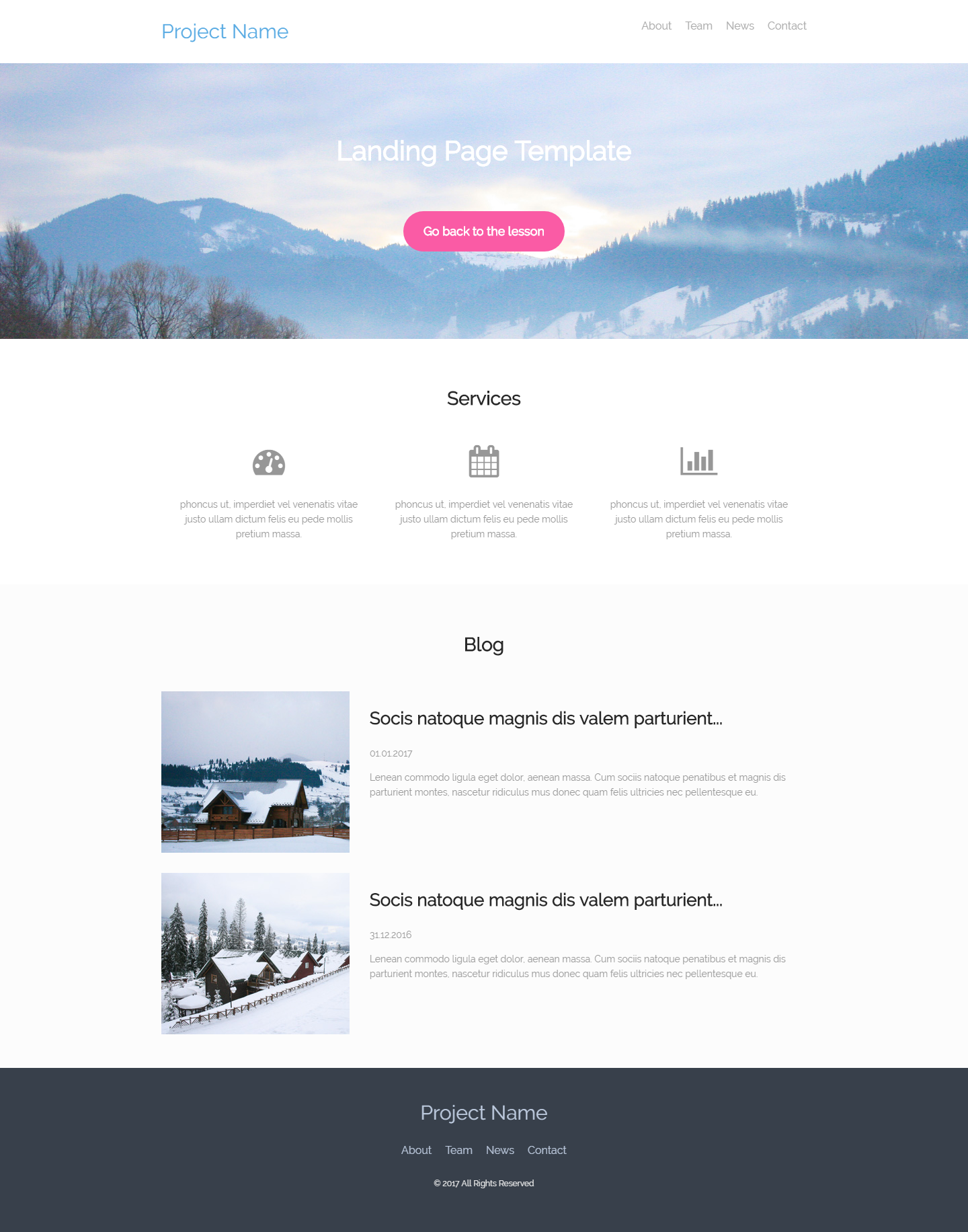
Введение

Открыв HTML-документ из архива, вы увидите готовый каркас посадочной страницы. В теге **<head>** мы подключили таблицы стилей, которые понадобятся вам для работы:

* **normalize.css** — файл для сброса стандартных стилей браузера (мы подробно рассказывали о нем в одном из уроков первой части учебника);
* шрифт Raleway от Google Fonts;
* иконочный шрифт FontAwesome;
* **style.css** — файл, куда вы будете записывать все стили для лендинга.

Тело документа разделено на секции — шапка сайта (header) с навигацией (nav), большой блок на всю ширину (jumbotron), блок сервисов (services), записи блога (blog) и подвал сайта (footer). Каждая секция содержит соответствующий контент.

**Макет страницы выглядит следующим образом:**

[[](http://m.siterx.ru/prodvizhenie/sozdanie-sajtov/blog/wp-content/uploads/landing-page-template.png)](http://m.siterx.ru/prodvizhenie/sozdanie-sajtov/blog/wp-content/uploads/landing-page-template.png)

Так должен выглядеть окончательный результат работы. Просмотрите файл **index.html** в браузере — пока что он выглядит не очень интересно, но совсем скоро вы это исправите собственноручно. Поехали!

Таблица стилей

Откройте файл **style.css** из папки «css». Вы увидите, что документ структурирован с помощью комментариев — для удобства мы разделили его на части в соответствии со структурой HTML-страницы. Записывайте каждый стиль в соответствующий раздел: стили для шапки сайта помещайте под комментарием **/\* Header \*/**, стили для блога — под комментарием **/\* Blog \*/**, базовые стили и стили для повторяющихся элементов страницы — под комментарием **/\* General \*/** и т. д.

Приступаем к работе

В процессе написания стилей мы будем двигаться сверху вниз, от шапки к подвалу, записывая стили для каждой секции. Но перед этим мы объявим несколько основных стилей для тегов и классов, которые будут использоваться постоянно.

Основные стили CSS

На нашей странице есть повторяющиеся элементы, а также теги, которым требуется задать определенный стиль, после чего он будет применен ко всем элементам сразу.

Начнем с основного текста на сайте. Согласно дизайну макета, основной (и единственный) шрифт на странице — это Raleway, а все параграфы **<p>** должны иметь размер шрифта 14 пикселей, цвет **#999** и высоту строки 1.6. Запишем эти данные в нашу таблицу стилей, в раздел General:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | body {  **font-family**: Raleway, Arial, Helvetica, sans-serif; /\* кроме Raleway добавьте                                                           \* несколько запасных шрифтов \*/  }  p {  **font-size**: 14px;  **color**: #999;  **line-height**: 1.6;  } |

Двигаемся дальше: нужны стили для заголовков. У нас есть один заголовок **<h1>**, а также несколько заголовков **<h2>** и **<h3>**. Согласно макету, заголовок h1 имеет размер шрифта 40 пикселей и белый цвет. Остальные заголовки имеют цвет **#282828**. Размер шрифта для заголовков второго уровня — 28 пикселей, а для h3 — 26 пикселей.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | h1 {  **font-size**: 40px;  **color**: #fff;  }  h2 {  **font-size**: 28px;  }  h3 {  **font-size**: 26px;  }  h2, h3 {  **color**: #282828;  } |

Помимо текста у нас есть еще несколько повторяющихся элементов. Например, класс **.container** — это блок, с ограниченной шириной в который помещается контент. Ширина нашего контента должна занимать 90% экрана, но при этом контейнер не должен растягиваться шире, чем 960 пикселей. Здесь нам пригодится недавно рассмотренное свойство **max-width**. Кроме того, этот контейнер должен размещаться по центру страницы, для чего мы воспользуемся свойством **margin**.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | .container {  **width**: 90%;  **max-width**: 960px;  **margin-left**: auto;  **margin-right**: auto;  } |

Следующие повторяющиеся элементы — блок с классом **.brand** и навигационное меню. Эти элементы присутствуют в хедере и футере. Они не будут выглядеть абсолютно идентично, однако часть стилей для них будут общими.

Итак, оба элемента **.brand** будут иметь одинаковый размер шрифта:

|  |  |
| --- | --- |
| 1  2  3 | .brand {  **font-size**: 30px;  } |

А оба навигационных меню будут горизонтальными и тоже будут иметь одинаковый размер шрифта. Кстати, для создания горизонтального меню мы задействуем свойство **float**, применив его к пунктам меню:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | .menu li {  **float**: left;  **font-size**: 16px;  **list-style**: none; /\* убираем стандартные маркеры списка \*/  }  .menu {  **padding-left**: 0; /\* убираем стандартный левый отступ списка \*/  **margin**: 0; /\* убираем стандартные поля списка \*/  } |

Уберем из навигации стандартное подчеркивание ссылок:

|  |  |
| --- | --- |
| 1  2  3 | .menu li a {  **text-decoration**: none;  } |

С основными стилями мы разобрались. Сохраните изменения в таблице CSS и обновите файл **index.html** в браузере. Вы уже можете видеть, как начала видоизменяться веб-страница. Теперь пора перейти к оформлению секций.

CSS для шапки сайта

Наш хедер будет иметь белый фон и отступ сверху в размере 30 пикселей. Запишем это в таблицу стилей:

|  |  |
| --- | --- |
| 1  2  3  4 | header {  **background-color**: #fff;  **padding-top**: 30px;  } |

Блок с названием проекта будет прилеплен к левой стороне контейнера, иметь цвет **#59abe3** и отступ внизу в 30 пикселей:

|  |  |
| --- | --- |
| 1  2  3  4  5 | header .brand {  **float**: left;  **color**: #59abe3;  **margin-bottom**: 30px;  } |

Обратите внимание, что мы записали стиль именно для того элемента **.brand**, который находится в шапке сайта, и не затрагивали **.brand**, расположенный в футере, поскольку он будет иметь другой цвет и расположение. В дальнейшем мы будем продолжать использовать преимущества селекторов потомков.

Навигационное меню мы приклеим к правой стороне шапки, добавим внизу поле высотой 30 пикселей, а также сделаем отступы по 20 пикселей слева для всех пунктов меню, чтобы они не сливались в одно целое:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | header .menu {  **float**: right;  **margin-bottom**: 30px;  }  header .menu li {  **padding-left**: 20px;  } |

Установим цвет текста **#aaa** для пунктов меню. Этот стиль мы применим именно к тегам **<a>**, а не к **<li>**, в противном случае мы не увидим результат. Дело в том, что тег **<a>** не наследует цвет текста от родителя, и ссылки продолжат отображаться в стандартном синем цвете.

|  |  |
| --- | --- |
| 1  2  3 | header .menu li a {  **color**: #aaa;  } |

Обновите страницу в браузере для просмотра изменений.

CSS для блока Jumbotron

Как правило, jumbotron (джамботрон) на сайте — это большой, привлекающий внимание блок, растянутый на всю ширину страницы. Фоном для нашего джамботрона послужит изображение в хорошем качестве, растянутое на всю ширину, а текстовое содержимое будет находиться в ограничивающем блоке **.container**.

Фоновое изображение вы найдете в папке «img». По правде говоря, мы немного забегаем вперед, поскольку тему о фоновых изображениях мы будем разбирать уже в следующем разделе. Но сейчас вы можете просто скопировать следующий CSS-код, посмотреть результат и попробовать самостоятельно понять, что и как в нем работает:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | .jumbotron {  **background-image**: url(../img/jumbotron.jpg);      background-**size**: cover;  **background-position**: top center;  **background-repeat**: no-repeat;  } |

Мы добавим еще несколько строк к стилю джамботрона — для создания вертикальных отступов по 80 пикселей и установки минимальной высоты в 250 пикселей, для выравнивания внутреннего содержимого по центру блока, а также для сброса обтекания, которое было задано навигационному меню:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | .jumbotron {  **background-image**: url(../img/jumbotron.jpg);      background-**size**: cover;  **background-position**: top center;  **background-repeat**: no-repeat;  **padding-top**: 80px;  **padding-bottom**: 80px;  **min-height**: 250px;  **text-align**: center;  **clear**: both;  } |

Стиль для заголовка джамботрона мы добавили ранее, поэтому сразу перейдем к созданию яркой кнопки, расположенной под ним. Если быть точнее, это обычная ссылка с классом **.button**, но ведь CSS позволяет нам творить чудеса!

Во-первых, поскольку тег **<a>** является строчным элементом, то первым делом применим к нему свойство **display** со значением **inline-block**. Таким образом кнопка будет отображаться как блок, а поведение останется, как у строки.

Текст кнопки будет белым и без подчеркивания, шрифт — жирный 18 пикселей, верхний margin — 40 пикселей, а цвет фона — **#22a7f0**. Чтобы это больше было похоже на кнопку, необходимо добавить отступы с четырех сторон — в нашем случае по 20 пикселей вертикально и по 30 пикселей горизонтально. Также мы округлим углы элемента с помощью уже известного нам свойства **border-radius**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | .jumbotron .button {  **display**: inline-block;  **color**: #fff;  **text-decoration**: none;  **font-weight**: bold;  **font-size**: 18px;  **margin-top**: 40px;  **background-color**: #fa5ba5;  **padding**: 20px 30px;      border-radius: 30px;  } |

CSS для блока Services

Данная секция будет включать в себя три блока, расположенных в один ряд. В этом нам снова поможет свойство **float**. Все блоки будут иметь одинаковую ширину, указанную в процентах, и в сумме они займут всю ширину контейнера. Разделив число 100 на 3, получаем 33,333333. Именно его мы записываем в стиль для класса **.service**. Дополнительно зададим каждому блоку внутренние отступы в размере 15 пикселей и внешние вертикальные отступы в 30 и 50 пикселей:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | .services .service {  **width**: 33.333333%;  **float**: left;  **padding-left**: 15px;  **padding-right**: 15px;  **margin-top**: 30px;  **margin-bottom**: 50px;  } |

Сейчас вы можете обновить страницу и увидеть, как третий блок съехал вниз. Догадаетесь, почему так произошло? Вспомните урок, где мы разбирали, как браузер вычисляет размеры элемента и что делает свойство box-sizing. Мы добавили нашим блокам свойство **padding**, и сумма их ширин превысила ширину контейнера, поэтому один из них не помещается в ряд. Добавьте к стилю элемента **.service** свойство **box-sizing** со значением **border-box**, после чего обновите страницу в браузере. Вы увидите, что теперь все блоки выстроились так, как надо.

Идем дальше. Добавим стили для самого блока **.services** — сделаем верхний отступ в 50 пикселей и центрируем всё текстовое содержимое:

|  |  |
| --- | --- |
| 1  2  3  4 | .services {  **padding-top**: 50px;  **text-align**: center;  } |

В качестве иконок блоков мы использовали иконочный шрифт FontAwesome. Мы не будем углубляться в эту тему (на официальном сайте FontAwesome есть очень простая и понятная инструкция по использованию), но мы изменим цвет иконок на более подходящий нам. Каждая иконка представляет собой векторный рисунок, перекрасить который не составит труда. Мы просто запишем стиль, как будто задаем цвет для текста:

|  |  |
| --- | --- |
| 1  2  3  4 | .services .fa {  **color**: #979797;  **margin-bottom**: 15px; /\* и добавим небольшой отступ \*/  } |

Обновите страницу в браузере и полюбуйтесь результатом. Нам осталось написать стиль для двух последних блоков.

CSS для блока Blog

Прежде всего необходимо отменить обтекание для блока **.blog**. Он будет иметь вертикальные внутренние отступы в 50 пикселей, а также легкий серый фон, в соответствии с макетом:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | .blog {  **clear**: both;  **padding-top**: 50px;  **padding-bottom**: 50px;  **background-color**: #fcfcfc;  } |

Выровняем заголовок этого блока по центру:

|  |  |
| --- | --- |
| 1  2  3 | .blog h2 {  **text-align**: center;  } |

Каждому блоку с классом **.post** присвоим внутренний отступ в 30 пикселей сверху и поведение строчно-блочного элемента:

|  |  |
| --- | --- |
| 1  2  3  4 | .blog .post {  **padding-top**: 30px;  **display**: inline-block;  } |

Для миниатюр постов зададим обтекание и внешний отступ 30 пикселей с правой стороны:

|  |  |
| --- | --- |
| 1  2  3  4 | .blog .post img {  **float**: left;  **margin-right**: 30px;  } |

После чего дополните стиль элементов **.post** строкой **clear:both**, благодаря которой каждый блок будет отображаться корректно и не съезжать.

CSS для футера

Для основного блока подвала понадобится сбросить обтекание, задать вертикальные отступы в 50 пикселей, выравнивание текста по центру и темно-серый фон:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | footer {  **clear**: both;  **padding-top**: 50px;  **padding-bottom**: 50px;  **text-align**: center;  **background-color**: #38404b;  } |

Установим цвет текста для элемента **.brand** (как уже упоминалось, он будет отличаться от стиля, применяемого к элементу **.brand** в шапке сайта):

|  |  |
| --- | --- |
| 1  2  3 | footer .brand {  **color**: #bcc9dd;  } |

Для пунктов меню в футере зададим отступы и цвет ссылок:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | footer .menu li {  **padding-left**: 10px;  **padding-right**: 10px;  }  footer .menu li a {  **color**: #bcc9dd;  } |

**Обратите внимание:** чтобы свойство **text-align:center**, примененное к тегу **<footer>**, повлияло на блок с меню, необходимо сделать его строчно-блочным:

|  |  |
| --- | --- |
| 1  2  3  4  5 | footer .menu {  **display**: inline-block;  **margin-top**: 30px; /\* также добавим отступы \*/  **margin-bottom**: 15px;  } |

И, наконец, напишем небольшой стиль для последнего элемента — **.copyright**:

|  |  |
| --- | --- |
| 1  2  3  4 | footer .copyright {  **font-size**: 12px;  **color**: #eee;  } |

Теперь обновите страницу и взгляните на результат вашей работы. Если вы всё сделали без ошибок, то у вас получилась веб-страница, которую вы видели на макете в начале данного урока.

Проверка адаптивности

Но это еще не всё. Давайте проверим, насколько наша верстка хорошо адаптируется под разные размеры экрана. Поиграйтесь с шириной окна браузера и понаблюдайте за поведением элементов. Уверены, вы обнаружили несколько недочетов. Давайте исправим их.

Первое, что бросается в глаза — это три блока **.service**, каждый из которых имеет ширину 33,333333% от ширины контейнера. На узких экранах текст в этих блоках становится трудночитаемым.

Это можно исправить, использовав медиазапрос (в предыдущем практическом уроке мы также его применяли, но подробное изучение медиазапросов будет немного позже). В медиазапросе можно указать, при какой ширине экрана какой стиль отображать.

Итак, наши блоки начинают выглядеть плохо на экранах шириной менее 576 пикселей (в основном, это телефоны в портретной ориентации). Запишем специальный стиль для данной ширины (поместите эту запись в самый низ таблицы стилей):

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | @media (**max-width**: 575px) {      .services .service {  **width**: 100%;  **float**: none;      }  } |

Теперь, если ширина экрана пользователя будет менее чем 576 пикселей, каждый блок **.service** будет иметь ширину 100% и не иметь обтекания.

Продолжаем проверять нашу верстку на адаптивность. Отчетливо видно, что на маленькие экраны нормально не помещаются блоки **.post**. Миниатюра поста чересчур большая для небольшого экрана и следующий за ней текст некрасиво обтекает ее. Исправим это следующим стилем (просто добавьте код к уже созданному медиазапросу):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | .blog .post img {  **float**: none;  **display**: block;  **margin-right**: auto;  **margin-left**: auto;  }  .blog .post {  **max-width**: 280px;  **margin-left**: auto;  **margin-right**: auto;  **display**: block;  } |

Напоследок добавим еще пару строк кода, которые помогут всем изображениям на странице стать адаптивными. Внимание: этот стиль добавляем в раздел General!

|  |  |
| --- | --- |
| 1  2  3  4  5 | img {  **display**: block;  **max-width**: 100%;  **height**: auto;  } |