**Адаптивный дизайн**

Для решения проблемы совместимости веб-страниц с самыми различными разрешениями самых различных устройств возникла концепция адаптивного (отзывчивого) дизайна (Responsive Web Design – RWD). Ее суть заключается в том, чтобы должным образом масштабировать элементы веб-страницы в зависимости от ширины экрана. Для разработки адаптивного макета страницы, нам потребуются три основных компонента:

• Гибкий макет на основе сетки (flexible, gridbased layout) – Организация контента без четко заданной ширины, которая состоит из пересекающихся вертикальных и горизонтальных полос. Сетка используется для структурирования контента в предсказуемой и последовательной манере, размер и позиционирование элементов можно настраивать.

• Гибкие изображения (flexible images) – Любой визуальный контент или медиа файл, который подстраивается под размер экрана пользователя. Обычно, это изображения, расположенные внутри гибкой сетки, которой задано CSS правило max-width: 100%. С точки зрения непрофессионала данный подход делает так, что изображения не могут быть больше сетки, внутри которой они находятся, но в то же время изображения могут менять свой размер, не теряя при этом соотношение сторон. Дабы избежать долгой загрузки, дизайнеры могут сжимать разрешение изображений для маленьких экранов. Другой способ – это установить ширину изображения в процентах от ширины страницы в целом.

• Медиа запросы (media queries) – Модуль CSS3, который контролирует применение стилей внутри медиа атрибута. Условием для применения различных стилей служит ширина экрана, его высота, ориентация и т.д.

Эти три краеугольных камня образуют основу того, что мы называем адаптивным веб-дизайном.

**Верстка адаптивной страницы**

В файле builder-home.psd имеется макет страницы для верстки.

Процесс верстки подразделяется на несколько этапов:

1. Создание структуры проекта при построении страницы

Вырезка изображений из psd-макета

1. Деление макета на отдельные блоки.

Для каждого блока:

3) создание html-структуры блока

4) стилизация блока

5) приведение блока к адаптивному виду.

Согласно psd-шаблона имеется 3 основных блока + кнопка, которые условно можно назвать как header, about-us и icons + button (рисунок 1).

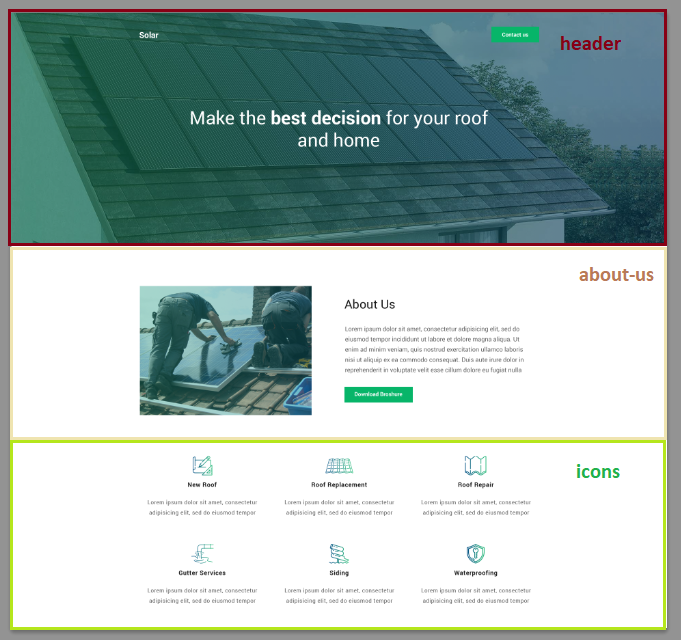


Рисунок 1

Создадим отдельные папки для хранения изображений и стилей, общий стиль страницы будем определять в файле core.css, стили блоков – в отдельных файлах с именами header.css, about-us.css, button.css и icons.css. Подключать их все к странице будем через файл style.css.

1. *Создание структуры проекта при построении страницы*

**Задание 1.** Создайте новую папку с названием Solar и внутри нее следующую структуру папок и файлов (рисунок 2):

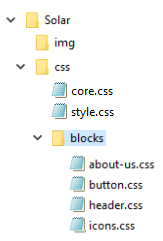


Рисунок 2

Создайте в папке Solar файл index.html следующего вида:

**Пример 1**

<!DOCTYPE HTML>

<head>

<title>Адаптивная страница №1</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="css\style.css" rel="stylesheet">

</head>

<body>

</body>

</html>

**Метатег viewport**

В примере1 обратите внимание на meta-тег meta name="viewport". Метатег viewport был разработан компанией Apple для того, чтобы указывать браузерам на то, в каком масштабе необходимо отображать пользователю видимую область веб-страницы. Атрибут name предназначен для того, чтобы указать браузеру, какую именно информацию о странице хотим ему сообщить. В данном случае эта информация касается viewport.

Контент (содержимое) этих сведений указывается в качестве значения атрибута content посредством пар ключ-значение, разделённых между собой запятыми.

Для адаптивного дизайна значения атрибута content viewport должно определяться 2 параметрами:

* width=device-width
* initial-scale=1

Параметр (width=device-width) отвечает за то, чтобы ширина видимой области веб-страницы равнялась ширине устройства (device-width).

1. *Извлечение изображений из файла шаблона*

**Задание 2**. Откройте файл для верстки - builder-home.psd.

Выбрав инструмент Перемещение с параметрами Автовыбор, Группы слоев, Найдите тот слой, который соответствует фоновому изображению в шапке. Уберите затенение. Создайте дубликат слоя с этим изображением в новом файле. Выполните обрезку прозрачных пикселей и сохраните файл bg. jpg для web в хорошем качестве в папке img. Еще одно фото сохраните аналогичным образом в файле img.jpg без затенения. Иконки сохраните в формате png в папке img, задав им имена icon1, icon2 и т.д. Всего 8 файлов.

1. *создание html-структуры блока* *шапки*

Согласно рисунку 3 все элементы страницы разбиваются на отдельные блоки для удобства их оформления стилями. Сине-фиолетовый контейнер содержит все остальные элементы и должен располагаться на всю доступную ему ширину. В шапке присутствует красный блок, растянутый по всей доступной ширине. Внутри него есть блок-обертка для заголовка, который состоит из верхней и нижней частей. Верхняя (помечена розовым) содержит логотип и кнопку. Нижняя – заголовок верхнего уровня.

**Задание 3**. Создайте вложенные блоки внутри тега body, задав классы согласно рисунку 3.

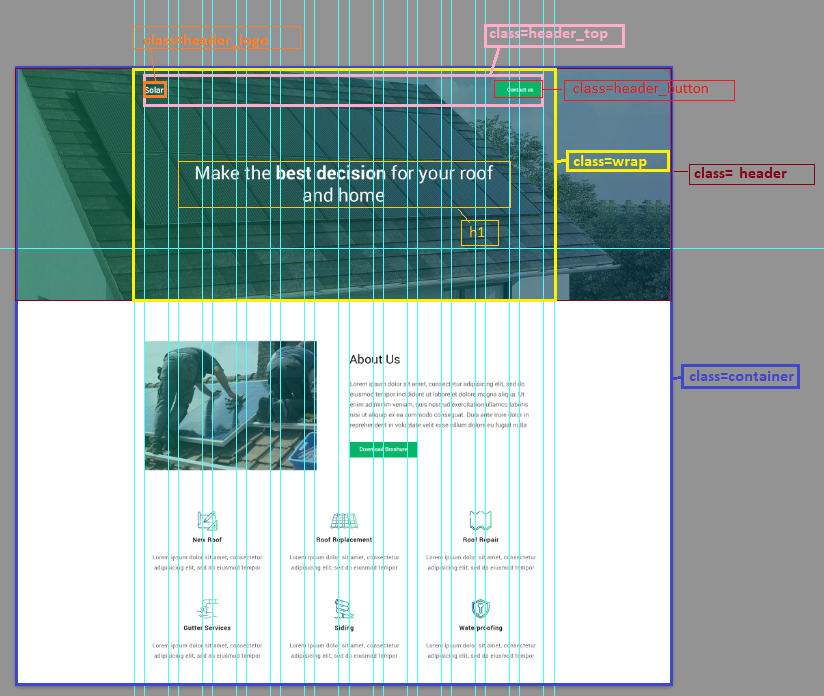


Рисунок 3

Результат должен соответствовать рисунку 4, при этом блок с классом header\_top должен включать блоки header\_logo и header\_top. В них логотип Solar задайте абзацем, а кнопку - ссылкой с href=”#”. Учтите так же, что заголовок должен быть полужирным начертанием не весь, а только фраза best decision. Для просмотра кода и его результатов нажмите в браузере F12.

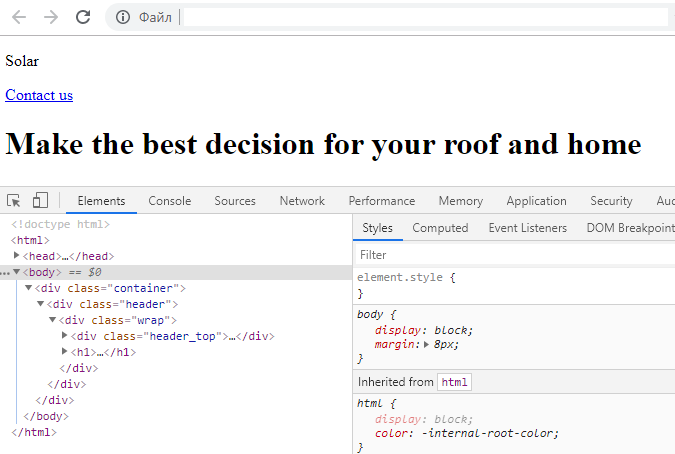


Рисунок 4

1. *Стилизация блока шапки*

**Использование правила import**

Правило **CSS import** позволяет импортировать внешние таблицы стилей не только на **HTML-страницу**, но и в другой документ css. Это упрощает управление файлами и их интеграцию.

Чтобы воспользоваться правилом **@import** для подключения файла /styles/default.css в **HTML-файле** нужно добавить следующий код в тег <style> или css-файл: @import url("/styles/default.css");

Если распределить **CSS** по отдельным файлам, то можно воспользоваться **@import** внутри основного стилевого файла. В нашем примере имеется основной стилевой файл style.css, в котором необходимо инструкцией подключить все остальные файлы стилей.

**Использование нестандартных шрифтов**

Для использования нестандартных шрифтов можно использовать коллекцию шрифтов Google с сайта <https://fonts.google.com/>. Найти подходящий шрифт, щелкнуть на знаке + справа от его названия, и в открывшемся окне на вкладке CUSTOMIZE выберите необходимые параметры шрифта, а на вкладке EMBED скопировать из поля @IMPORT код для подключения шрифта к странице (указывается отдельной строкой в файле css), а из поля Specify in CSS скопировать задание стилей для конкретного элемента веб-страницы.

**Задание 4**. Общие для всей страницы стили задайте в файле core.css и style.css:

а) Отступы (margin) и поля (padding) для селектора \* задайте нулевыми.

б) задайте максимальную ширину классу container 100%, а минимальную (делаем адаптивную верстку) ширину – 320 px.

в) включите сетку и для блока с классом wrap измерьте его ширину в psd-макете, полученное значение в px укажите как значение максимальной ширины блока, задайте ему нулевые поля по вертикали и 15 px – по горизонтали и отцентрируйте весь блок по горизонтали.

г) в psd-макете выясните название нестандартного шрифта, используемого на странице, и подключите его с сайта [https://fonts.google.com](https://fonts.google.com/). Скопированную инструкцию @import разместите первой в файле style.css. А сам шрифт назначьте для тега body в файле core.css.

д) с помощью правил @import подключите все остальные файлы стилей в файле style.css

После подключения полученных стилей страница должна выглядеть следующим образом (рисунок 5):

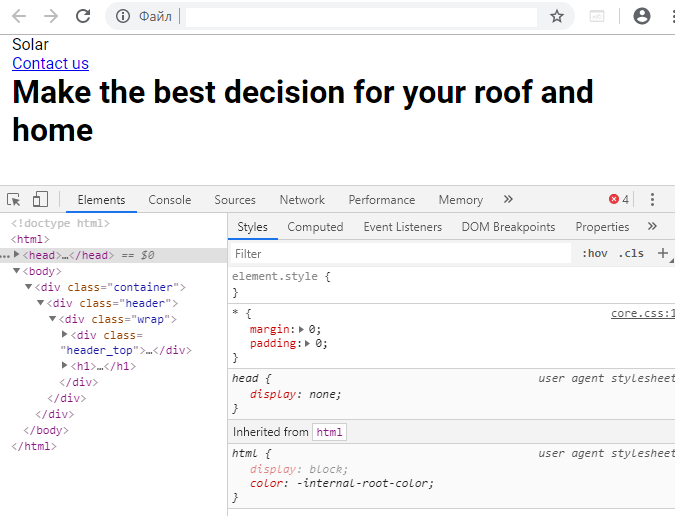


Рисунок 5

# Относительные единицы vh, vw, vmin, vmax

В CSS3 были добавлены новые относительные единицы измерения, такие как vh, vw, vmin, vmax. Эти единицы вычисляются относительно размеров окна браузера (см. таблицу 1)

Таблица 1

|  |  |
| --- | --- |
| **Единица** | **Описание** |
| vh | Эквивалентно 1% высоты окна браузера. |
| vw | Эквивалентно 1% ширины окна браузера. |
| vmin | Эквивалентно 1% меньшего размера окна браузера по высоте или ширине. |
| vmax | Эквивалентно 1% большего размера окна браузера по высоте или ширине. |

**Единицы vh и vw**

Приёмы адаптивного веб-дизайна базируются на использовании процентных значений. Но проценты далеко не лучшее решение для каждого случая, так как они вычисляются относительно размеров ближайшего родительского элемента. Поэтому, лучше использовать высоту и ширину окна браузера единицы измерения vh и vw. Например, если высота окна браузера равна 900px, то 1vh будет равен 9px. Аналогично, если ширина окна браузера равна 1600px, то 1vw будет равен 16px.

**CSS flexbox**

CSS flexbox (Flexible Box Layout Module) – модуль макета гибкого контейнера – представляет собой способ компоновки элементов. В основе flexbox лежит идея оси.

Flexbox состоит из flex-контейнера – родительского контейнера и flex-элементов – дочерних блоков. Дочерние элементы могут выстраиваться в строку или столбик, а оставшееся свободное пространство распределяется между ними различными способами.

Модуль flexbox позволяет решать следующие 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 их переполнения;
* Создавать колонки одинаковой высоты;
* Создавать прижатый к низу страницы [подвал сайта](https://html5book.ru/goto/https:/codepen.io/html5book/pen/QMLRLR).

Если элементу задать свойство display: flex;, то его дочерние элементы станут flex-элементами и упорядочивать их расположение внутри этого родительского блока можно с помощью следующих свойств justify-content (таблица 2)– для выравнивания по горизонтали, и align-items (таблица 3) – для выравнивания по вертикали.

**Выравнивание элементов по горизонтали свойство justify-content**

Таблиц 2 – Значения свойства justify-content

|  |  |
| --- | --- |
| flex-start | Значение по умолчанию. Flex-элементы позиционируются от начала flex-контейнера. |
| flex-end | Flex-элементы позиционируются относительно правой границы flex-контейнера. |
| center | Flex-элементы выравниваются по центру flex-контейнера. |
| space-between | Flex-элементы выравниваются по главной оси, свободное место между ними распределяется следующим образом: первый блок располагается в начале flex-контейнера, последний блок – в конце, все остальные блоки равномерно распределены в оставшемся пространстве, а свободное пространство равномерно распределяется между элементами. |
| space-around | Flex-элементы выравниваются по главной оси, а свободное место делится поровну, добавляя отступы справа и слева. |
| initial | Устанавливает значение свойства в значение по умолчанию. |
| inherit | Наследует значение свойства от родительского элемента. |

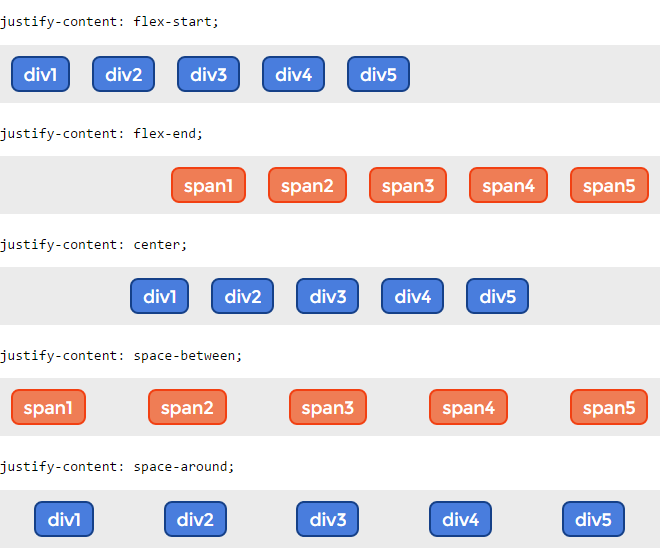
Результат действия свойства с различными значениями на рисунке 6.

Рисунок 6 – Выравнивание элементов и распределение свободного пространства с помощью свойства justify-content

**Выравнивание элементов по вертикали align-items**

Свойство выравнивает flex-элементы, в том числе и анонимные flex-элементы по перпендикулярной оси (по высоте).

Таблица 3 – Значения свойства **align-items**

|  |  |
| --- | --- |
| stretch | Значение по умолчанию. Flex-элементы растягиваются, занимая все пространство по высоте. |
| flex-start | Flex-элементы выравниваются по левому краю flex-контейнера относительно верхнего края блока-контейнера. |
| flex-end | Flex-элементы выравниваются по левому краю flex-контейнера относительно нижнего края блока-контейнера. |
| center | Flex-элементы выравниваются по центру flex-контейнера. |
| baseline | Flex-элементы выравниваются по базовой линии. |
| initial | Устанавливает значение свойства в значение по умолчанию. |
| inherit | Наследует значение свойства от родительского элемента. |

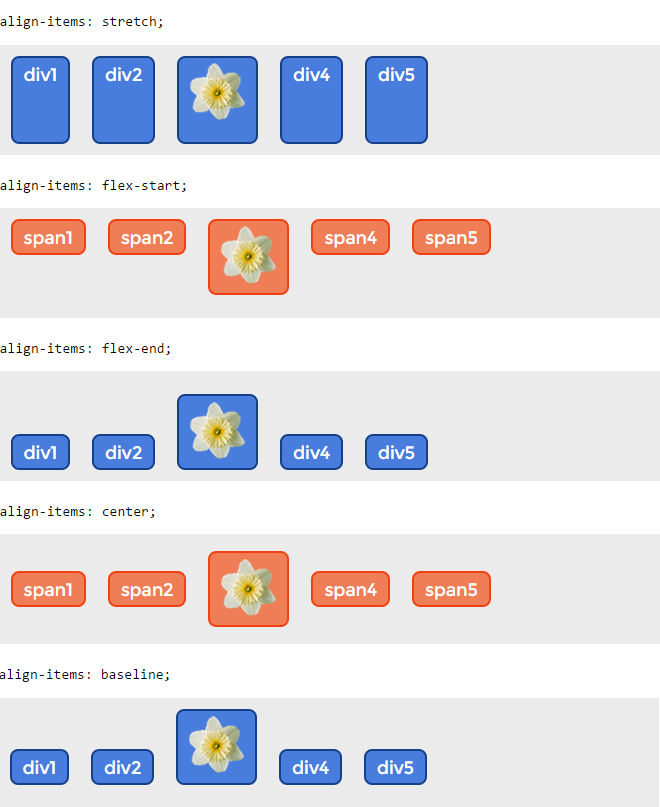
Результат действия свойства с различными значениями на рисунке 7.

Рисунок 7 – Выравнивание элементов в контейнере по вертикали

Таким образом, чтобы выровнять дочерние элементы для flex-контейнера по вертикали и горизонтали внутри родительского блока нужно ему указать свойства:

display: flex;

align-items: flex-start;

justify-content: space-between

**Задание 5**. Задайте стили для шапки страницы в файле header.css:

а) в качестве фона для класса header по центру задайте файл ../../img/bg.jpg без повторений и свойство background-size в значение cover.

б) к классу wrap в index.html добавьте класс header\_wrap, описав для него в файле header.css стили:

.header\_wrap{

позиционирование: относительное;

высота: 100vh

}

г) нижнее поле (padding-bottom) для класса header измерьте в psd-макете – расстояние между двумя изображениями.

д) сделайте с помощью стилей класс header\_top (файл header.css) flex-контейнером, задайте для его дочерних элементов выравнивание по горизонтали, таким образом, чтобы два дочерних элемента логотип и кнопка былии прижаты к краям контейнера (см. psd-макет), а по вертикали были выровнены по центру. Также в psd-макете измерьте расстояние от кнопки до верха страницы и задайте измеренное значение в качестве вертикального поля класса header\_top.

е) c помощью psd-макета посмотрите стили для логотипа, вызвав свойства текста для надписи Solar – цвет, размер, межстрочный интервал, начертание (свойству, ответственному за полужирное начертание задайте значение 700). Определите эти стили для тега p, находящегося внутри класса .header\_top.

ж) на странице две кнопки, поэтому кнопку будем оформлять отдельным css-файлом button.css. Для этого в class="header\_button" добавьте еще одно имя класса button и для ссылки внутри этого класса задайте стили, считанные из свойств текста Contact us и задайте цвет фона для кнопки, воспользовавшись пипеткой для его копирования в в psd-макете. Нужно убрать подчеркивание, задать цвет, размер, межстрочный интервал, полужирное начертание для текста, а также цвет его фона. Задайте свойство display: inline-block; и измерьте расстояние от начала и конца надписи до границ кнопки, это будет значение левого и правого полей, поля сверху и снизу подберите, исходя из общей высоты кнопки в макете.

з) задайте стили для заголовка Make the best decision for your roof and home, просмотрев их в psd-макете: цвет, размер, межстрочный интервал (60px), начертание (свойству, ответственному за полужирное начертание задайте значение 400 – не жирный) и выравнивание текста – по центру.

и) для тега span внутри заголовка h1 задайте полужирное начертание свойству, ответственному за полужирное начертание задайте значение 700)

к) для заголовка h1 для выравнивания внутри родительского блока задайте параметры: max-width: 970px;

width: 100%;

position: absolute;

left: 50%;

top: 50%;

padding: 0 15px;

box-sizing: border-box;

transform: translate(-50%, -50%);

Проверьте соответствие макета и отображаемой страницы в части заголовка.

**CSS3-медиазапросы**

В 2001 году в HTML4 и CSS2 была введена поддержка аппаратно-зависимых таблиц стилей, позволившая создавать стили и таблицы стилей для определенных типов устройств. Для этого были введены медиа-запросы.

**Что такое медиа-запрос**

В общем случае медиа-запрос состоит из ключевого слова, описывающего тип устройства (необязательный параметр) и выражения, проверяющего характеристики данного устройства. Медиа-запрос является логическим выражением, которое возвращает истину или ложь.

Общий синтаксис:

@media (условие) {описание стилей}

Медиа-запросы могут быть добавлены следующими способами:

* 1. С помощью HTML:

<link rel="stylesheet" media="screen and (color)" href="example.css">

* 1. С помощью правила @import внутри <style> или css-файла:

@import url(color.css) screen and (color);

Например, непосредственно в коде страницы:

<style>

@media (max-width: 600px) {

#sidebar {display: none;}

}

</style>

И, например, внутри таблицы стилей style.css:

@media (max-width: 600px) {

#sidebar {display: none;}

}

Предыдущие два примера означают, что стиль будет действовать только в случае ширины экрана менее 600px.

Рассмотрим, какие характеристики позволяют проверять медиа-запросы(таблица 4)

|  |  |
| --- | --- |
| Таблица 4 – Характеристики носителя | |
| **Параметр** | **Описание** |
| width | Проверяет ширину области просмотра. Значения задаются в единицах длины, px, em и т.д., например, (width: 800px). Обычно для проверки используются минимальные и максимальные значения ширины.min-width применяет правило если ширина области просмотра больше значения, указанного в запросе, max-width – ширина области просмотра меньше значения, указанного в запросе. |
| height | Проверяет высоту области просмотра. Значения задаются в единицах длины, px, em и т.д., например, (height: 500px). Обычно для проверки используются минимальные и максимальные значения высоты.min-height применяет правило если высота области просмотра больше значения, указанного в запросе, max-height – высота области просмотра которого меньше значения, указанного в запросе. |
| aspect-ratio | Проверяет соотношение ширины к высоте области просмотра. Широкоэкранный дисплей с соотношением сторон 16:9 может быть помечен как (aspect-ratio: 16/9).min-aspect-ratio проверяет минимальное соотношение, max-aspect-ratio – максимальное соотношение ширины к высоте области просмотра. |
| orientation | Проверяет ориентацию области просмотра. Принимает два значения: (orientation: portrait) и (orientation: landscape). |
| resolution | Проверяет разрешение экрана (количество пикселей). Значения также могут проверять количество точек на дюйм (dpi) или количество точек на сантиметр (dpcm), например, (resolution: 300dpi).min-resolution проверяет минимальное разрешение экрана, max-resolution – максимальное. |
| color | Проверяет количество бит на каждый из цветовых компонентов устройства вывода. Например, (min-color: 4) означает, что экран конкретного устройства должен иметь 4-битную глубину цвета. min-color проверяет минимальное количество бит, max-color – максимальное количество бит. |
| color-index | Проверяет количество записей в таблице подстановки цветов. В качестве значения указывается положительное число, например, (color-index: 256).  min-color-index проверяет минимальное количество записей, max-color-index – максимальное количество записей. |

1. *Приведение блока шапки к адаптивному виду*

При уменьшении экрана при просмотре нашей страницы некоторые детали (отступы, размеры шрифта и изображений) приходится адаптировать под новые размеры для более удобного просмотра.

**Задание 6.** Используя медиа-запросы, установите следующие стили в файле header.css:

а) при ширине экрана от 980px и меньше нижний отступ для класса .header задайте в 60 px

б) при ширине экрана от 768px и меньше для заголовка h1 внутри класса .header задайте размер шрифта 34px, межстрочный интервал – 40px.

Откройте страницу в браузере и проверьте, сужая размер окна браузера до нужных размеров.

в) проблема: при уменьшении высоты просмотра, наш заголовок съезжает на место блока .header\_top. Поэтому создайте еще один медиа-запрос для высоты экрана 480px и менее со стилями для .header\_wrap{height:auto;} и стилями для .header h1{position:static;

left: auto;

top: auto;

transform: translate(0, 0);}

Откройте страницу в браузере и проверьте, уменьшая высоту окна браузера до нужных размеров.

г) добавьте в медиа-запросе верхнее поле в размере 24px, а нижнее – 150px.