HyperText Markup Language

HTML&CSS

Урок 7

Адаптивный дизайн

Оглавление

1. Что такое адаптивный дизайн? 3
2. Принципы адаптивного дизайна 7
3. Метатег viewport.....11
4. Медиа-запросы 14
Домашнее задание.....44

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

1. Что такое адаптивный дизайн?

В современном мире существует огромное количество устройств, с помощью которых мы выходим в интернет: компьютеры, планшеты, смартфоны, телевизоры и даже часы. И пользователю будет не удобно просматривать веб-страницу, если она на смартфоне откроется в таком же виде, как и на ноутбуке: элементы интерфейса будут слишком малы или появится горизонтальная прокрутка. Пользователь скорее всего уйдет с подобного сайта, если не применить к последнему адаптивный дизайн.

Давайте рассмотрим пример как бы отображался известный поисковик Google, если бы у него не было адаптивного дизайна.

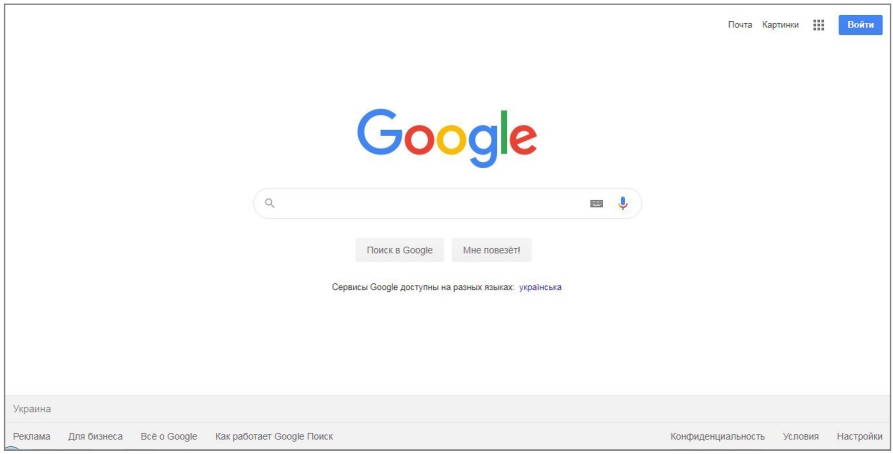

Рисунок 1. Отображение поисковика Google на ноутбуке

На рисунке 1 все элементы удобно расположены для использования и занимают всю видимую часть окна браузера на ноутбуке.

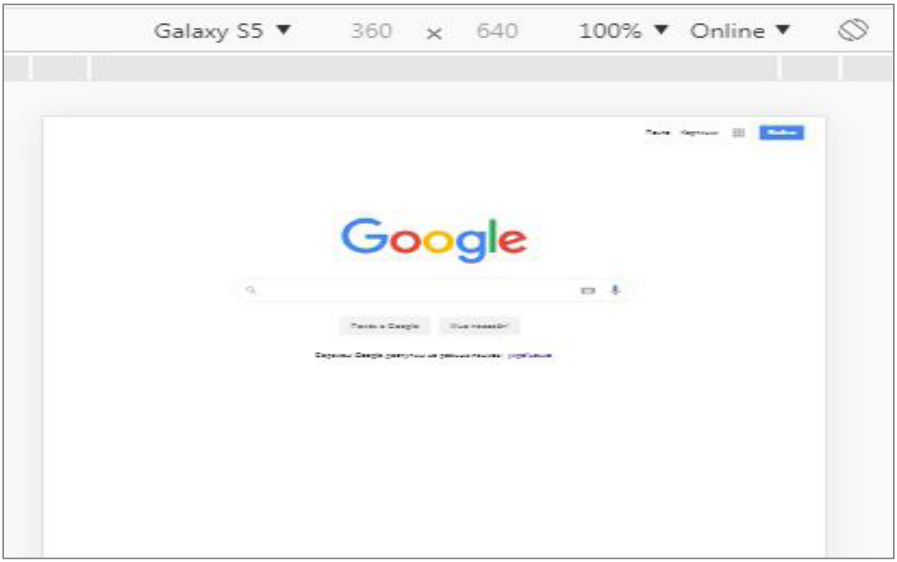

Рисунок 2. Отображение поисковика Google на Galaxy S5 без адаптивности

На рисунке 2 видно, что элементы страницы стали очень маленькие и не удобны ни для просмотра, ни для использования.

На рисунке 3 видно, как адаптивный дизайн перестроил страницу поисковика: все элементы снова приняли удобное расположение и размеры, соответствующие устройству.

Адаптивный дизайн — это дизайн, который предоставляет правильное отображение веб-страницы на различных устройствах и плавно изменяет элементы страницы при изменении размера окна браузера.

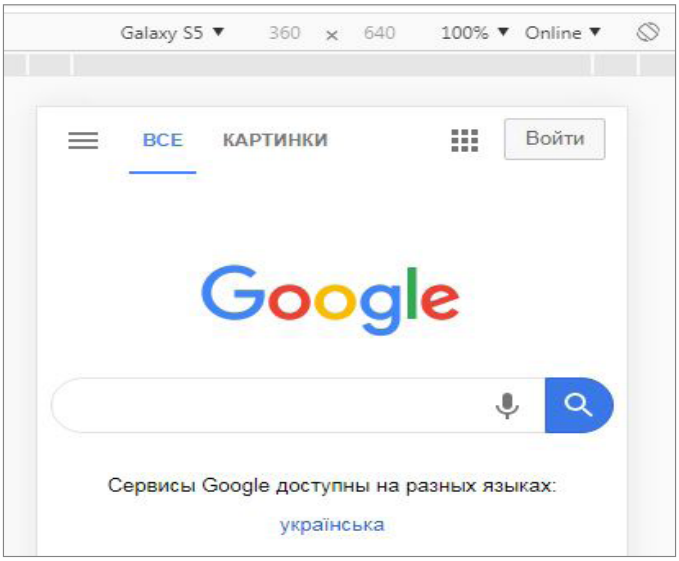

Рисунок 3. Отображение поисковика Google на Galaxy S5 с адаптивностью

В недалеком прошлом было принято создавать отдельную «мобильную» версию веб-сайта, которая на самом деле была копией контента с другим отображением элементов, что приводило к проблемам в продвижении таких сайтов. С приходом понятия адаптивного дизайна отдельные версии сайта для различных устройств создавать не нужно, потому что при одном и том же контенте мы получаем удобное расположение элементов на мобильных устройствах.

Преимущества адаптивного дизайна:

- Отсутствует необходимость создавать отдельную версию сайта для конкретного устройства.
- Универсальное представление веб-страниц для различных устройств.

- Все страницы доступны по одному url адресу, что избавляет от проблем в продвижении сайта.
- Удобство использования интерфейса веб-страниц независимо от гаджета.

Недостатки адаптивного дизайна:

- Медленная скорость загрузки сайта из-за большого веса. Независимо от устройства грузится полная версия сайта.
- Более сложная верстка веб-страниц, так как нужно учесть тонкости отображения на всех видах устройств.
- Невозможность «отключить» мобильное отображение на устройстве с маленьким разрешением, необходимо открывать страницу на другом устройстве с большим разрешением.
- Более сложный и длительный процесс тестирования сайта.

Однако, даже учитывая недостатки, адаптивный дизайн необходим, так как количество мобильных пользователей и разнообразие гаджетов растет с каждым днем.

2. Принципы адаптивного дизайна

1. **Использование относительных единиц измерения.** Для того, чтобы элементы изменялись плавно в зависимости от ширины окна браузера, необходимо использовать относительные единицы измерения для указания ширины, высоты, внутренних и внешних отступов, размера шрифта. К относительным единицам относятся: %, em, ex, vh, vw, vmin.
2. **Применение граничных значений для ширины контейнера.** На маленьком устройстве контент на ширину всего окна смотрится хорошо, а вот такое же отображение контента на широкоформатном мониторе уже будет вызывать дискомфорт во время просмотра. Поэтому рекомендовано использовать граничные значения для ширины/высоты в абсолютных величинах, а именно в пикселах. Для этого используются свойства: min-width/min-height и max-width/max-height.
3. **Использование структуры в виде сетки.** В CSS3 существуют различные инструменты, которые позволяют строить гибкую структуру для расположения элементов. Такими инструментами являются модуль Flexbox и сетка Grid Layout.
4. **Использование медиа-запросов для перестройки отображения элементов страницы.** Для того, чтобы страницы не просто плавно сжимали контент

в зависимости от ширины окна браузера, а перестраивали содержимое под гаджет для удобного просмотра, необходимо прописывать контрольные точки. Контрольные точки — это физический параметр устройства, по которому определяется текущее представление. Устанавливаются контрольные точки с помощью медиа-запросов.

5. **Начинать верстку с мобильного отображения и постепенно продвигаться к широкоформатному или наоборот.** Принято начинать верстку адаптивной веб-страницы, начиная с маленьких устройств, так как в них меньше элементов и в основном простое и лаконичное отображение контента, постепенно переходя к большим размерам экрана. Однако, можно действовать и наоборот, двигаться от широкоформатного устройства к мобильному представлению.
6. **Использование системных шрифтов.** Конечно, вы будете использовать в дизайне различные шрифты, которые представлены в макете. Так же нужно помнить, что шрифт, подгружаемый с ресурса в интернете замедляет его загрузку. Системные шрифты загружаются мгновенно, чем существенно ускоряют загрузку веб-страницы.
7. **Скрытие или замена элементов на различных устройствах.** На мобильных устройствах часто скрывают элементы, которые не несут информативности (например, баннер с рекламой стороннего ресурса), либо прячут их за границы экрана, чтобы можно было развернуть их в любой удобный момент (например, фильтры товара).

8. **Адаптация графического и видео контента.** Для того, чтобы страницы подгружались быстрее, необходимо использовать изображения, адаптированные под текущее отображение. Например, фоновые изображения меньшего размера для мобильных устройств. В тех случаях, когда это возможно, использовать векторный формат изображений вместо растрового.

Примеры адаптивного дизайна можно посмотреть на ресурсе Media Queries (mediaqueri.es), где представлена галерея веб-сайтов с адаптивным дизайном.

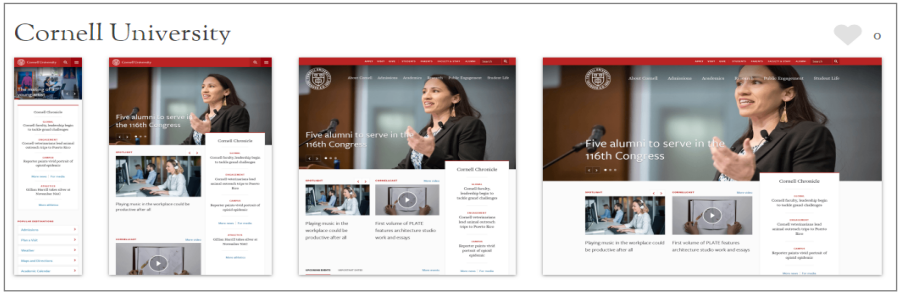

Рисунок 4

На рисунке видно, что элементы сайта перестраиваются в зависимости от устройства и его размера.

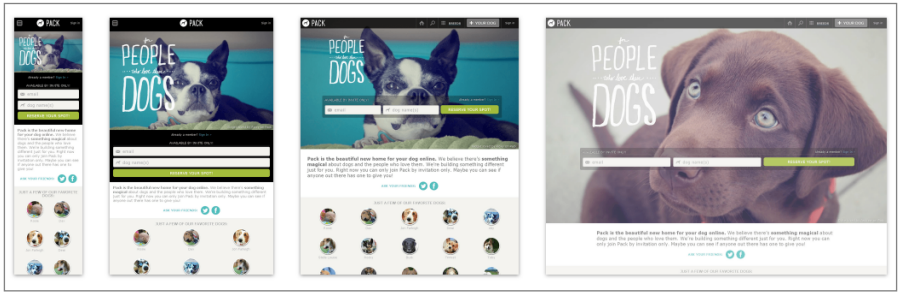

Рисунок 5

Элементы интерфейса перестраиваются под устройство для более удобного использования, так же подгружаются различные изображения для более быстрой загрузки сайта.

3. Метатег viewport

Viewport — это видимая область окна браузера, в которую последний вписывает веб-страницу. Каждый браузер определяет ширину viewport по-своему. Например, у Safari — 980 пикселей, у IE — 1024 пикселя. В среднем, ширина колеблется около 1000 пикселей, потому что считается, что веб-страницы предназначены для десктопных мониторов.

Процесс отображения страницы состоит из следующих этапов: браузер получает страницу с сервера, задает ей размеры своей ширины viewport, а затем пропорционально сжимает страницу до размеров отображаемого устройства. Для того, чтобы браузер не масштабировал страницу, приняв ее ширину за ширину viewport установленную по умолчанию в его настройках, необходимо использовать метатег viewport.

Метатеги предназначены для указания информации для браузеров и поисковых систем. Метатег viewport указывает браузеру в каком масштабе необходимо отображать видимую часть страницы на различных устройствах.

Таблица 1

Параметр	Значение	Описание
width	Целое число в пикселях или значение device-width, которое равно ширине экрана в пикселях CSS при масштабе 100%	Задает ширину области viewport. Ширина в пикселях CSS — это не физическое разрешение экрана, а величина регламентирующая размер пикселя.

Параметр	Значение	Описание
		Необходима для того, чтобы при большой плотности пикселей, элементы выглядели одинаково
height	Целое число в пикселях или значение device-height, которое равно высоте экрана в пикселях CSS при масштабе 100%	Задает высоту области viewport
initial-scale	Вещественное число от 0.1 и выше	Задает коэффициент масштабирования начального размера viewport. (1.0 — отсутствие масштабирования)
user-scalable	no/yes	Запрещает/разрешает пользователю масштабировать страницу
minimum-scale	Вещественное число от 0.1 и выше	Задает минимальный масштаб размера viewport. (1.0 — отсутствие масштабирования)
maximum-scale	Вещественное число от 0.1 и выше	Задает максимальный масштаб размера viewport. (1.0 — отсутствие масштабирования)

В таблице 1 представлен перечень параметров и их значений, которые может принимать метатег viewport.

Для того, чтобы браузер понимал, что страница является адаптированной под различные устройства, необходимо прописать метатег viewport со следующими значениями:

```
<meta name="viewport" content="width=device-width,
      initial-scale=1">
```

Значения, указанные в атрибуте content, являются значениями по умолчанию. Их следует изменять, если ваш сайт должен иметь, например, минимальную ширину не меньше 450 пикселей. Тогда значение метатега

```
<meta name="viewport" content="width=450,
      initial-scale=1">
```

будет фактически задавать эту минимальную ширину для viewport. Если же экран устройства будет шириной более 450 пикселей, то браузер будет расширять область просмотра, а не увеличивать ее.

Второй пример используют в тех случаях, когда настройки масштаба должны оставаться неизменными при переключении ориентации:

```
<meta name="viewport" content="initial-scale=1,
      maximum-scale=1">
```

В спецификации CSS3 определение тега <meta name = "viewport"> является ненормированным, то есть на текущий момент для этого тега нет установленного стандарта.

4. Медиа-запросы

Медиа-запрос по сути является условной конструкцией, которая запрашивает у устройства, которое отображает веб-страницу, его характеристики и выполняет набор стилевых правил, если полученные характеристики устройства соответствуют заданным в условии текущего медиа-запроса.

Медиа-запросы используются в адаптивном дизайне в тех случаях, когда необходимо применить разные css-стили для отображения на различных устройствах с учетом их характеристик. Например, на рисунке 6 видно, как перестраивается страница от устройства к устройству.

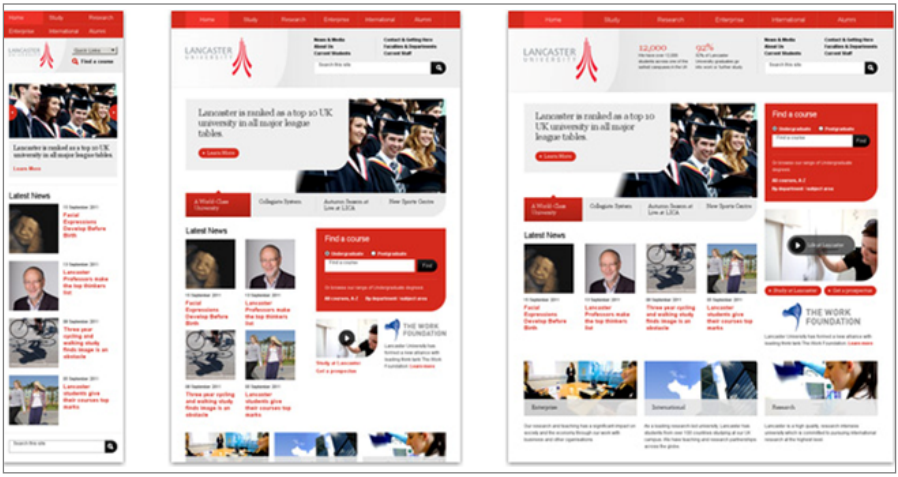

Рисунок 6

Обратите внимание на элемент выполняющий поиск по сайту. На десктопном отображении он находится

в верхнем правом углу страницы, оставаясь на том же месте при выводе результатов поиска. В мобильной же версии отображения этот элемент перемещается вниз страницы, так как это самое удобное место для быстрого ввода искомой информации.

Медиа-запрос состоит из типа устройства (необязательный параметр) и технических характеристик данного устройства.

Медиа-запрос можно применить следующими способами:

1. С помощью тега link:

```
<link rel="stylesheet" media="screen and
      (min-width:900px)" href="width_900.css">
```

2. С помощью правила @import:

```
@import url(width_900.css) screen and (min-width:900px);
```

3. С помощью правила @media, указанного внутри тега style либо в стилевом файле:

```
@media screen and (min-width:900px){
   /* стили для указанного типа устройства и его
      характеристик
   */
}
```

Типы устройств:

- all — все устройства (используется по умолчанию);
- print — режим предварительного просмотра перед печатью;

- screen — экраны мониторов;
- speech — синтезаторы речи.

В CSS2 перечень типов устройств был больше. Туда входили отдельно проекторы, телевизоры, планшеты и другие устройства, которые в современном стандарте приняты как устаревшие.

Ниже представлена таблица 2 с характеристиками устройств, которые являются обязательной частью медиа-запроса.

Таблица 2

Наименование	Описание
aspect-ratio	Отношение ширины к высоте. Например: (aspect-ratio: 12/5)
color	Количество бит на компонент цвета
color-gamut	Проверяет цветовую гамму, которая поддерживается устройством: rgb(color-gamut:srgb), p3(color-gamut:p3), BT.2020(color-gamut:rec2020)
color-index	Проверяет использует ли устройство таблицу соответствия цветов. Значение: целое число
grid	Проверяет является ли устройство вывода сеточным или растровым. Если устройство вывода представляет сетку (например, терминал или дисплей телефона с одним фиксированным шрифтом), то значение будет равным 1. В противном случае будет 0
height	Высота области видимости. Задается в абсолютных и относительных единицах измерения
monochrome	Количество бит на пиксель монохромного устройства. Задается целым числом
orientation	Ориентация устройства: портретная (orientation:portrait) или альбомная (orientation: landscape)

Наименование	Описание
overflow-block	Описывает поведение устройства, когда идет переполнение контентом по вертикальной оси в режиме горизонтальной записи и по горизонтальной оси при вертикальной записи. Значения: нет (overflow-block:none), скролл (overflow-block:scroll), с подгрузкой (overflow-block: optional-paged), постранично (overflow-block: paged)
overflow-inline	Описывает поведение устройства, когда идет переполнение контентом по горизонтальной оси в режиме горизонтальной записи и по вертикальной оси при вертикальной записи. Значения: нет (overflow-inline:none), скролл (overflow-block:scroll)
resolution	Разрешение экрана — количество пикселей на дюйм (dpi) или сантиметр (dpcm)
scan	Проверяет процесс рендеринга устройств: чересстрочный (scan:interlace) и прогрессирующий (scan:progressive)
update	Проверяет возможность обновлять содержимое после его визуализации (например, анимацию css): нет (update:none), медленно (update:slow) и быстро (update:fast)
width	Ширина области видимости. Задается в абсолютных и относительных единицах измерения

Медиа запрос может содержать комплексную проверку характеристик устройства. Комбинированный медиа запрос создается с помощью логических операторов:

1. and — объединяет несколько медиа-функций в один медиа-запрос. Запрос выполняется только в том случае, если все медиа-функции соответствуют характеристикам устройства. Например:

```
@media (min-width:320px) and (max-width:480px){}
```

Этот запрос выполнится только в случае, если ширина экрана устройства находится в диапазоне от 320 до 480 пикселей.

2. Запятая — объединяет несколько медиа-запросов в одно правило. Каждый запрос обрабатывается отдельно от других, таким образом стили применяются, если хотя бы один запрос соответствует характеристикам устройства. Например:

```
@media screen and (aspect-ratio: 16/9),
       screen and (aspect-ratio: 16/10){}
```

Этот запрос выполнится для мониторов, у которых пропорциональное отношение ширине к высоте будет равно 16/9 или 16/10.

3. not — используется для инвертирования медиа-запроса. Например:

```
@media not (color){}
```

Этот запрос выполнится для устройств, которые не поддерживают цветность.

4. only — используется для применения стиля только в случае соответствия всего запроса. Например, скрывает стили для старых браузеров.

Примеры формирования медиа-запросов с учетом типа и характеристики устройства:

- **Ширина/высота** — это самые популярные варианты медиа-запросов, которые считывают ширину или высоту устройства:

Устройство и параметры	Запрос
Смартфоны с шириной экрана в диапазоне от 320 до 480 пикселей	@media only screen and (min-width:320px) and (max-width:480px) { /* стили */ }
Смартфоны с максимальной высотой экрана 600 пикселей	@media only screen and (max-height:600px) {/* стили */ }

- **Ориентация** (альбомная и портретная) — происходит проверка ориентации устройства, если такова имеется и включена:

Устройство и параметры	Запрос
Смартфоны с альбомной ориентацией	@media screen and (orientation: landscape) { /* стили */ }
Предварительный просмотр на принтере с портретной ориентацией	@media print and (orientation: portrait) {/* стили */ }

- **Цвет** — происходит проверка устройства на возможность отображать различные цвета:

Устройство и параметры	Запрос
Устройства не поддерживающие различные цвета, например, черно-белый принтер	@media not (color) { /* стили*/ }
Устройства, поддерживающие 4-битный цвет или меньше	@media (max-color: 4) {/* стили */}

- **Пропорции** — проверяет пропорции экрана монитора, а именно отношение ширины устройства к его высоте:

Устройство и параметры	Запрос
Широкоформатные мониторы с пропорциональным отношением ширины к высоте равным 16/9	@media screen and (aspect-ratio: 16/9){ /* стили*/ }

Устройство и параметры	Запрос
Квадратные мониторы с пропорциональным отношением ширины к высоте равным 1/1	@media screen and (aspect-ratio: 1/1){ /* стили*/ }

- **Разрешение** — проверяет разрешение устройства — количество пикселей на дюйм или сантиметр:

Устройство и параметры	Запрос
Устройства с разрешением 150dpi (количество пикселей на дюйм)	@media (max-resolution: 150dpi){ /* стили*/ }

- **Комплексные медиа** запросы используются в тех случаях, когда нужно проверить несколько параметров или применить одинаковые стили к различным устройствам и их техническим характеристикам:

Устройство и параметры	Запрос
устройство имеет минимальную высоту 680 пикселей или является экранным устройством в портретном режиме	@media (min-height: 680px), screen and (orientation: portrait) { /* стили*/ }

Давайте рассмотрим на примере небольшой страницы как реализуются медиа запросы на практике.

Создадим страницу, в которой будут отображаться основные элементы: header, main на три колонки, footer. В качестве содержимого пропишем названия тегов и их положение на десктопном отображении страницы.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content=
          "width=device-width, initial-scale=1.0">
    <title>Media</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <header>header top</header>
        <main>
            <aside>aside left</aside>
            <section>section center</section>
            <aside>aside right</aside>
        </main>
        <footer>footer bottom</footer>
    </div>
</body>
</html>
```

Вот что получилось:

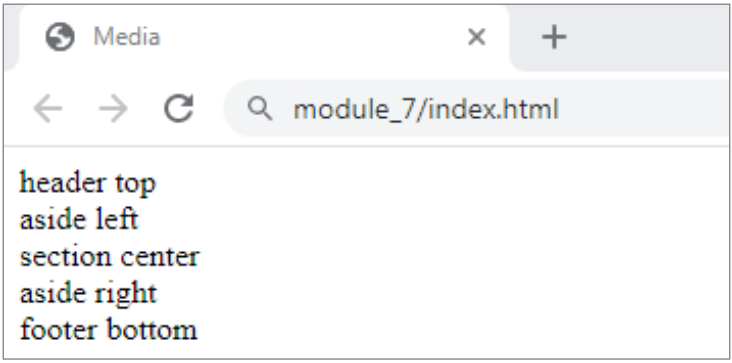

Рисунок 7

На рисунке выше видно, что все элементы располагаются друг под другом, так как являются блочными тегами. Займемся стилевым оформлением.

```
/* обратимся ко всем элементам, чтобы убрать отступы,
   которые назначит браузер по умолчанию */
*{
    padding: 0;
    margin: 0;
}
/* для основного контейнера будем использовать
   фиксировано-резиновую верстку */
.container{
    /* устанавливаем основную ширину равной ширине
       видимой части окна браузера*/
    width: 100vw;
    /* устанавливаем минимальную ширину, до которой
       можно сжимать контейнер, если ширина устройства
       будет меньше - появится горизонтальный скролл */
    min-width: 320px;
    /* устанавливаем максимальную ширину, до которой
       можно расширять контейнер, если ширина
       устройства будет больше - по бокам появятся
       отступы */
    max-width: 1200px;
    /* для того, чтобы контейнер находился всегда
       по центру по горизонтали указываем значения
       для внешних отступов */
    margin: auto;
    /* в случае, если контента будет мало, чтобы
       страница полностью заполнила всю высоту окна
       браузера, устанавливаем минимальную высоту*/
    min-height: 100vh;
    /* в случае, если контента будет много, чтобы
       появился вертикальный скролл, устанавливаем
       значение высоты равной содержимому*/
    height: auto;
    /* для более простого расположения элементов
       на странице, воспользуемся технологией
       flexBox */
    display: flex;
```

```
    /* направление основной оси меняем на вертикальное */
    flex-direction: column;
}
/* каждому элементу контейнера будем устанавливать
   фоновой цвет, чтобы было нагляднее */
header{
    background-color: pink;
    /* зададим размер header по основной оси */
    flex-basis: 10vh;
}
/* main - основной элемент страницы является
   одновременно и flex-родителем и flex-контейнером */
main{
    /* чтобы заполнять все пространство по основной
       оси основным элементом прописываем свойство
       flex */
    flex: 1 1 auto;
    display: flex;
}
main>aside{
    background-color: cyan;
    /* зададим размер левой части по основной оси
       в процентах */
    flex-basis: 20%;
}
main>section{
    background-color: coral;
    /* чтобы заполнять все пространство по основной оси
       основным элементом прописываем свойство flex */
    flex: 1 1 auto;
}
main>section+aside{
    background-color: lime;
    /* зададим размер левой части по основной оси
       в процентах */
    flex-basis: 20%;
}
```

```
footer{
    background-color: gray;
    /* зададим размер footer по основной оси */
    flex-basis: 5vh;
}
```

На текущий момент, мы получили следующее отображение для широкоформатного монитора:

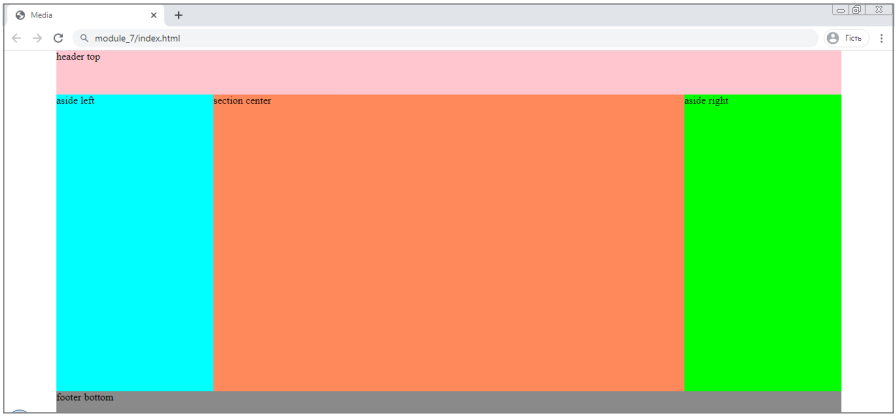

Рисунок 8

Переходим к созданию медиа-запросов. Так как мы изначально строили страницу для десктопного отображения, то теперь пойдем по убывающей постепенно приближаясь к мобильному устройству. Для этого создадим два медиа-запроса для отображения на «квадратных» мониторах и на смартфонах.

```
/* запрос для экранов с максимальной шириной
   960px и меньше*/
@media screen and (max-width:960px){
    main>section+aside{
```

```
        /* спрячем правый элемент на всех устройствах,
           с шириной 960px и меньше*/
        display: none;
    }
    main>aside{
        /* зададим размер левой части по основной оси
           в процентах*/
        flex-basis: 30%;
    }
    main>section{
        /* зададим размер центральной части
           по основной оси в процентах */
        flex-basis: 70%;
    }
}
/* запрос для экранов с максимальной шириной 570px
   и меньше */
@media screen and (max-width:570px){
    main{
        /* меняем направление основной оси элемента
           отображающего основной контент */
        flex-direction: column;
    }
    main>aside{
        /* зададим размер видимому элементу aside
           по основной оси */
        flex-basis: 20vh;
    }
    main>section{
        /* зададим размер основному контенту
           по основной оси*/
        flex-basis: auto;
        /* изменим порядок отображения элементов так,
           чтобы контент находился выше*/
        order:-1;
    }
}
```

Итого, мы получили еще два вида отображения. Для квадратных мониторов:

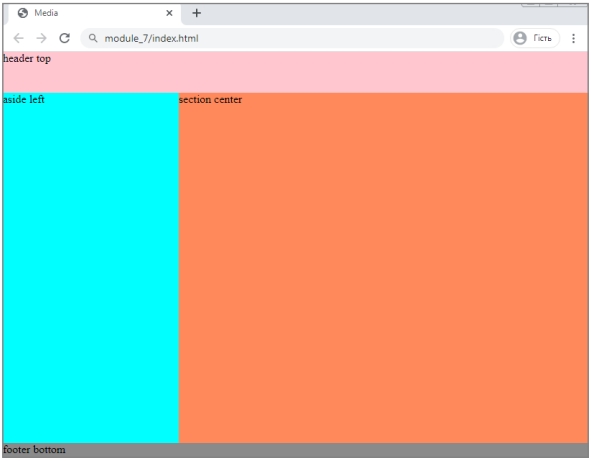

Рисунок 9

А также для смартфонов:

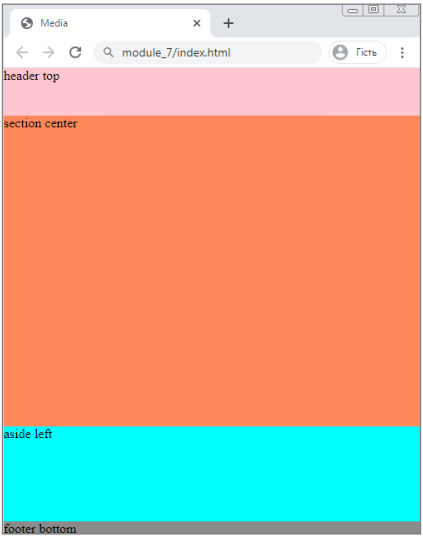

Рисунок 10

› *С полным кодом примера можно ознакомиться в прикрепленных к уроку файлах или просмотреть в CodePen по ссылке.*

Рассмотрим пример создания небольшой страницы новостного сайта. На широкоформатных мониторах контент будет отображаться в трех колонках, на квадратных — в двух и на мобильных — в одной колонке.

На рисунке ниже представлено отображение на широкоформатном мониторе.

Рисунок 11

Создадим страницу *index.html* со следующим содержанием:

1. Шапка сайта включает название сайта и главное меню;
2. Основная часть сайта состоит из:
   - колонки последних новостей, находящейся слева,
   - основного контента — статьи с заголовком, изображением и иконками социальных сетей, находящегося посередине,
   - колонки цитат, находящейся справа.

3. Подвал сайта содержит знак копирайта и текущий год.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content=
        "width=device-width, initial-scale=1.0">
   <title>Media</title>
   <!-- подключаем шрифт Bitter с ресурса
        fonts.googleapis.com -->
   <link href="https://fonts.googleapis.com/
              css2?family=Bitter&display=swap"
         rel="stylesheet">
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <div class="container">
      <header>
         <h1>Media</h1>
         <nav>
            <a href="">Home</a>
            <a class="active" href="">News</a>
            <a href="">Gallery</a>
            <a href="">Contact</a>
         </nav>
      </header>
      <main>
         <aside>
            <div>
               <h3>Last News</h3>
               <article class="news">
                  <strong>business</strong>
                  <h2>
                    <a href="">Conseptur eos
                       ipsam possimus</a>
                  </h2>
                 <em>by Modi Cumque</em>
```

```
            <p>Mollitia modi cumque architecto
               commodi illum, conseq sed nihil
               error  aliquam  accusamus.
            </p>
            <strong>2 hours ago</strong>
         </article>
         <article class="news">
            <strong>art</strong>
            <h2>
               <a href="">Lorem architecto
                  commodi illum</a>
            </h2>
            <em>by Timus Onihil</em>
            <p>Dolore ipsam possimus mollitia
               modi cumque architecto commodi
               illum, consequuntur!</p>
            <strong>4 hours ago</strong>
         </article>
      </div>
   </aside>
   <section>
      <article>
         <h2>Conseptur eos ipsam possimus</h2>
         <em>by Modi Cumque</em>
         <img src="images/urbanization.jpg" alt="">
         <p>Facilis minus alias consequuntur
            esse autem ducimus quaerat, delectus
            obcaecati quidem voluptatibus corrupti
            ex, aliquam, error quisquam dicta ut.
            Eum quasi quidem fugit corporis
            labore velit voluptatibus consectetur,
            laudantium cupiditate?</p>
         <p>Tempora earum iure corporis iste
            ea voluptates impedit doloribus
            minima, error nisi.Eum cum sunt
            inventore nesciunt quod dicta?
            Nostrum consectetur odit sit nulla
```

```
            ad odio debitis rem sapiente animi
            impedit aut tenetur autem vitae
            inventore numquam dolore iste quos,
            architecto sed? </p>
         <p>Repellat quisquam numquam a
            inventore dolores, modi quod autem
            voluptatibus! Lorem ipsum dolor sit
            amet consectetur adipisicing elit.
            Architecto, et. Tempora, doloremque
            inventore? Minima fugiat quis quas
            ipsam voluptates adipisci?</p>
         <div class="social">
            <a class="fb" href=""></a>
            <a class="tw" href=""></a>
            <a class="inst" href=""></a>
         </div>
      </article>
   </section>
   <aside>
      <div>
        <h3>Osed accusamus</h3>
        <figure>
          <img src="images/unnamed.png" alt="">
          <figcaption>
            <p>Consectetur eos sunt, dolore
            ipsam possimus mollitia modi cumque
            architecto commodi illum!</p>
            <em>by Modi Cumque</em>
          </figcaption>
            </figure>
            <figure>
              <img src="images/unnamed.png" alt="">
              <figcaption>
                <p>Odio in, delectus commodi
                   sapiente nostrum ratione
                   porro laborum facere quasi
                   cumque!</p>
                <em>by Timus Onihil</em>
              </figcaption>
```

```
                    </figure>
                </div>
            </aside>
        </main>
        <footer>
            <p>&copy; 2020</p>
        </footer>
    </div>
</body>

</html>
```

Сейчас все элементы располагаются на странице последовательно в порядке их описания, как показано на рисунке ниже:

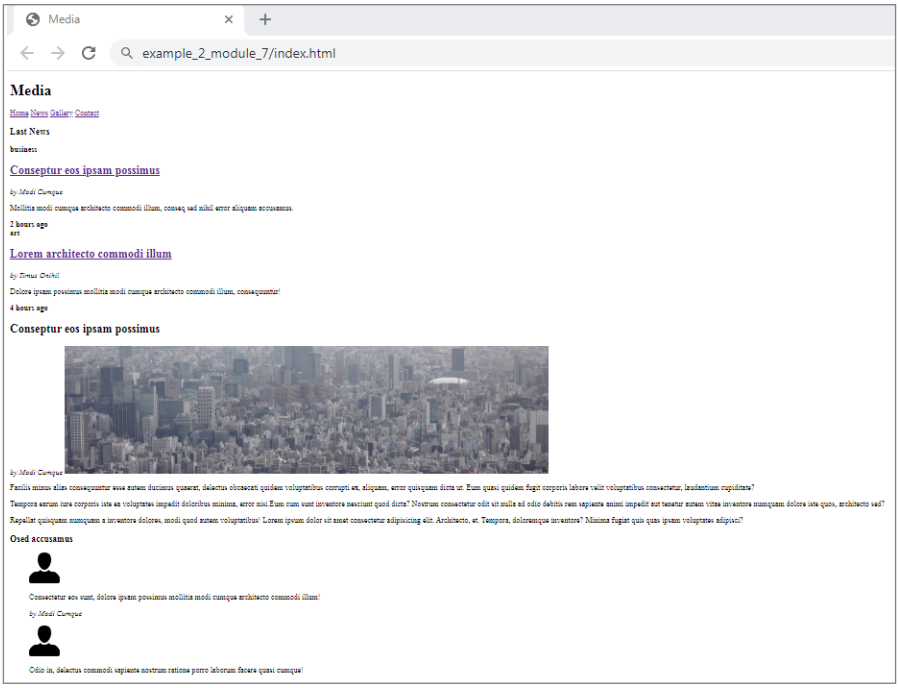

Рисунок 12

Приступим к созданию стилевого файла. Сначала выполняем стилевое оформление для основного отображения, в нашем случае это широкоформатные мониторы.

```
/* обратимся ко всем элементам, чтобы убрать отступы,
   которые назначит браузер по умолчанию */
*{
    padding: 0;
    margin: 0;
}

html {
    /* зададим основной размер шрифта, для того,
       чтобы пользоваться относительной единицей rem */
    font-size: 16px;
}

body{
    /* скроем контент, превышающий ширину окна
       браузера */
    overflow-x: hidden;
    /* устанавливаем основной шрифт контента -
       системный - это позволит быстрее загрузить
       страницу */
    font-family: Helvetica, Tahoma, Verdana, sans-serif;
    /* Единица rem задаёт размер относительно размера
       шрифта элемента <html> */
    font-size: 1rem;
    color: rgb(75, 75, 75);
}

h1,h2,h3,a{
    /* шрифт для заголовков и ссылок будет 'Bitter',
       который мы подключили с помощью тега <link> */
    font-family: 'Bitter', serif;
    font-weight: 400;
    color: rgb(105, 105, 105);
}
```

```
/* для основного контейнера будем использовать
   фиксировано-резиновую верстку*/
.container{
    /* устанавливаем основную ширину равной ширине
       видимой части окна браузера*/
    width: 100vw;
    /* устанавливаем минимальную ширину, до которой
       можно сжимать контейнер, если ширина
       устройства будет меньше - появится
       горизонтальный скролл */
    min-width: 320px;
    /* устанавливаем максимальную ширину, до которой
       можно расширять контейнер, если ширина
       устройства будет больше - по бокам появятся
       отступы */
    max-width: 1200px;
    /* для того, чтобы контейнер находился всегда
       по центру по горизонтали указываем значения
       для внешних отступов */
    margin: auto;
    /* в случае, если контента будет мало, чтобы
       страница полностью заполнила всю высоту окна
       браузера, устанавливаем минимальную высоту */
    min-height: 100vh;
    /* в случае, если контента будет много, чтобы
       появился вертикальный скролл, устанавливаем
       значение высоты равной содержимому */
    height: auto;
    /* для более простого расположения элементов
       на странице, воспользуемся технологией
       flexBox */
    display: flex;
    /* направление основной оси меняем на вертикальное */
    flex-direction: column;
}

/* каждому элементу контейнера будем устанавливать
   фоновой цвет, чтобы было нагляднее*/
header{
```

```
    /* зададим размер header по основной оси */
    flex-basis: 10vh;
    /* элементы шапки сайта располагаются слева
       на право */
    display: flex;
    /* между элементами задаем отступ */
    justify-content: space-between;
    /* отступ справа и слева зависит от ширины окна
       браузера */
    padding: 0 3vw;
}

header h1{
    /* выравниваем содержимое дочернего элемента
       по вертикали */
    align-self: center;
}

header nav{
    /* выравниваем содержимое дочернего элемента
       по вертикали */
    align-self: center;
    /* для того, чтобы легко расположить ссылки,
       воспользуемся снова технологией flexBox */
    display: flex;
}

/* стилевое оформление ссылок основного меню */
header nav a{
    display: block;
    margin-left: 2vw;
    padding: 0.5vw 1vw;
    /**/
    font-size: 1.1rem;
    text-decoration: none;
    border: 2px solid transparent;
}
```

```
header nav a:hover,
header nav a.active{
    border-bottom: 2px solid rgba(105, 105, 105, 0.5);
}

/* main - основной элемент страницы является
   одновременно и flex-родителем и flex-контейнером */
main{
    /* чтобы заполнять все пространство по основной
       оси основным элементом прописываем свойство
       flex */
    flex: 1 1 auto;
    display: flex;
    /* дочерние элементы располагаются слева на право */
    justify-content: space-between;
    padding-top: 2vh;
}

main>aside{
    /* зададим размер левой части по основной оси
       в процентах*/
    flex-basis: 20%;
}

main>aside>div{
    /* задаем отступы в процентах от ширины
       родительского элемента*/
    padding: 3% 10%;
}

main>aside>div>*{
    border-bottom: 2px solid rgba(105, 105, 105, 0.3);
    margin-bottom: 2vh;
}

main>aside>div>h3{
    text-transform: capitalize;
}
```

```
.news{
    font-size: 0.9rem;
    padding-bottom: 5px;
}

.news>*{
    display: block;
    padding-bottom: 7px;
}

.news>h2>a{
    font-size: 1.1rem;
    text-decoration: none;
}

.news>strong,
.news>em{
    color: rgb(155, 155, 155);
    font-size: 0.7rem;
}

.news>strong{
    text-transform: uppercase;
}

main>section{
    /* чтобы заполнять все пространство по основной
       оси основным элементом прописываем свойство
       flex */
    flex: 1 1 55%;
}

main>section>article{
    padding: 1% 5%;
}

main>section>article>*{
```

```
    display: block;
    padding-bottom: 1.5vh;
}

main>section>article>h2{
    text-transform: uppercase;
}

main>section>article>em{
    color: rgb(155, 155, 155);
    font-size: 0.9rem;
}

main>section>article>img{
    /* задаем изображению ширину 100% от ширины
       родительского элемента */
    width: 100%;
}

.social{
    text-align: right;
}
/* социальные сети подключаем используя изображение,
   состоящее из трех иконок социальных сетей.
   В этом случае загружается одно изображение вместо
   трех и, следовательно, скорость загрузки сайта
   выше. */

.social>a{
    /* меняем отображение ссылки со строчного
       на строчно-блочный */
    display: inline-block;
    /* размер ссылки равен размеру одной иконки */
    width:32px;
    height: 32px;
    /* в случае, если путь к изображению не содержит
       пробелов или специальных символов, можно
       прописывать его без кавычек */
```

```
    background-image: url(images/social_icons.png);
    background-repeat: no-repeat;
    /* устанавливаем размер так, чтобы отображалась
       только одна иконка */
    background-size: 300%;
    opacity: 0.7;
}

/* изменяем позиционирование фонового изображения
   в соответствии с классом для соцсети */
.social>a.fb{
    background-position: 0 0;
}

.social>a.tw{
    background-position: 50% 0;
}

.social>a.inst{
    background-position: 100% 0;
}

main>section+aside{
    /* зададим размер левой части по основной оси
       в процентах*/
    flex-basis: 20%;
}

main>section+aside figure *{
    display: block;
    padding-bottom: 7px;
}

main>section+aside figure img{
    opacity: 0.5;
    width: 32px;
    height:32px;
}
```

```
main>section+aside figure figcaption{
    font-size: 0.9rem;
}

main>section+aside figure figcaption em{
    color: rgb(155, 155, 155);
    font-size: 0.7rem;
}

footer{
    /*зададим размер footer по основной оси*/
    flex-basis: 5vh;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
```

Сейчас страница выглядит так:

Рисунок 13

Приступим к созданию медиа-запросов:

```
/* запрос для экранов с максимальной шириной 960px
   и меньше */
@media screen and (max-width:960px){
    main>section+aside{
        /* спрячем правый элемент на всех
           устройствах, с шириной 960px и меньше */
        display: none;
    }
    main>aside{
        /* зададим размер левой части по основной оси
           в процентах */
        flex-basis: 30%;
    }
    main>section{
        /* зададим размер центральной части
           по основной оси в процентах*/
        flex-basis: 70%;
    }
}
/* запрос для экранов с максимальной шириной 570px
   и меньше*/
@media screen and (max-width:570px){
    header{
        flex-basis: auto;
         /* меняем направление основной оси шапки
            сайта */
        flex-direction: column;
    }
    main{
        /* меняем направление основной оси элемента
           отображающего основной контент */
        flex-direction: column;
    }
    main>aside{
        /* зададим размер видимому элементу aside
           по основной оси */
        flex-basis: 20vh;
    }
```

```
    main>aside>div {
        padding: 3% 5%;
    }
    main>section{
        /* зададим размер основному контенту
           по основной оси*/
        flex-basis: auto;
        /* изменим порядок отображения элементов так,
           чтобы контент находился выше */
        order:-1;
    }
}
```

Теперь для квадратных мониторов страница приобретает следующий вид:

Рисунок 14

И для мобильных устройств страница выглядит так:

Рисунок 15

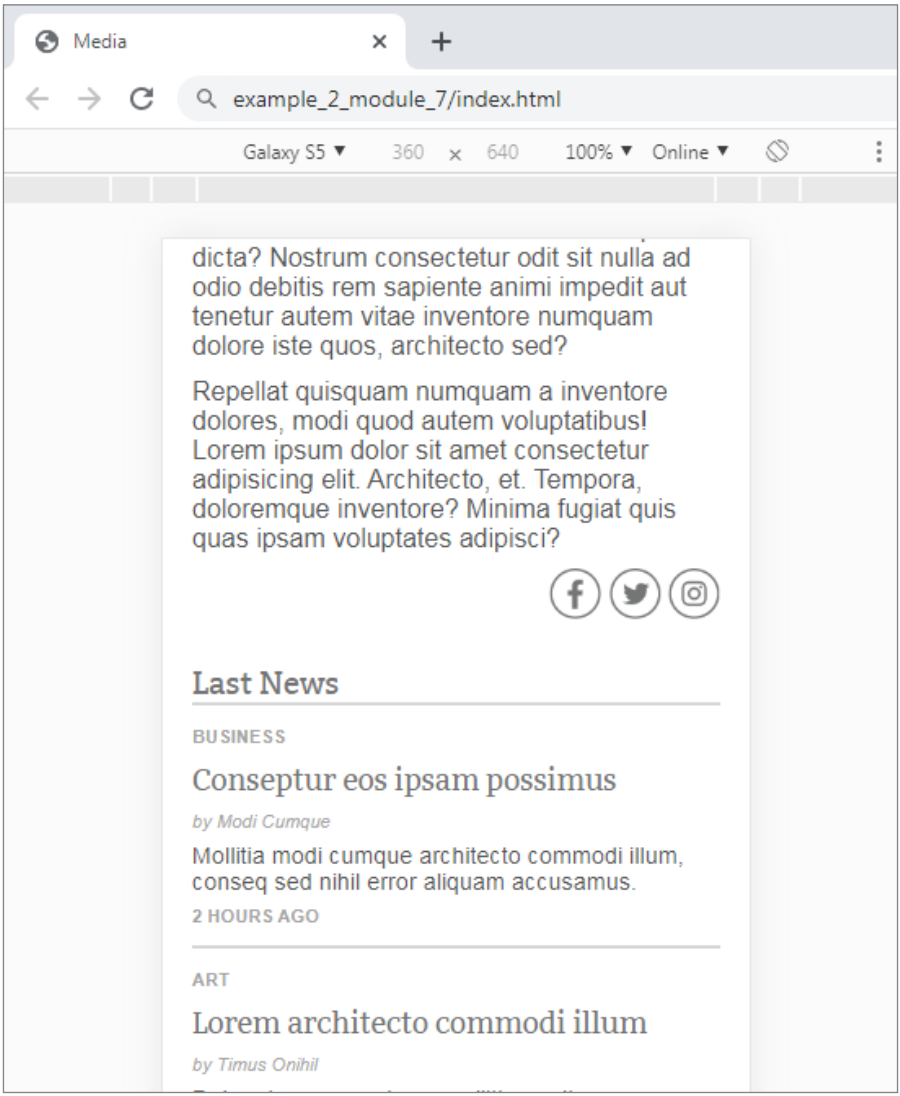

Рисунок 16

- *С полным кодом примера можно ознакомиться в прикрепленных к уроку файлах или просмотреть в CodePen по ссылке.*

Домашнее задание

Выполнить верстку страницы по макету, представленному на рисунке ниже.

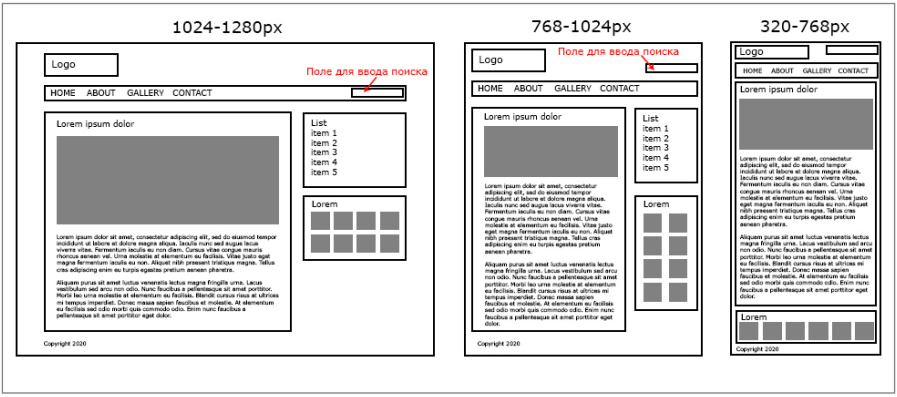

Рисунок 17