Методические указания по выполнению лабораторных работ по дисциплине «Веб-дизайн и шаблоны проектирования веб-приложений»

Введение

Учебная дисциплина «Веб-дизайн и шаблоны проектирования веб-приложений» является составной частью цикла дисциплин по информационным технологиям, изучаемых студентами специальности 1-40 05 01 «Информационные системы и технологии (по направлениям)». Занимая важное место в общепрофессиональной подготовке студентов, учебная дисциплина «Веб-дизайн и шаблоны проектирования веб-приложений» обеспечивает подготовку специалиста, владеющего фундаментальными знаниями и практическими навыками в области веб-дизайна и разработки шаблонов для проектирования веб-приложений.

Лабораторные занятия предназначены для закрепления и более глубокого изучения определенных аспектов лекционного материала на практике.

Материалы к лабораторным работам располагаются в Интернете по адресу https://github.com/sergei-tsarik/polessu-web.

1. Инструменты для разработки веб-приложений

В качестве основных инструментов для разработки веб-приложений предлагается использовать следующие программы и сервисы:

- Visual Studio Code – программа для написания HTML, CSS и JS.
- GitHub Pages – сервер для размещения веб-сайта.

1.1. Visual Studio Code

Visual Studio Code (VS Code) – текстовый редактор, разработанный Microsoft. Позиционируется как редактор кода для кроссплатформенной разработки веб-приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности настройки: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом.

Установочные файлы доступны на официальной странице загрузки https://code.visualstudio.com/download.

После установки VS Code рекомендуется выполнить настройку программы и установить дополнительные расширения (Extensions).

По умолчанию создаваемые и редактируемые файлы не сохраняются автоматически и каждый раз, когда внесены изменения в содержание файлов их необходимо сохранять, используя команду меню File > Save, или комбинацию горячих клавиш CTRL + S. Для включения режима «Автосохранение файла» (Auto Save) необходимо выполнить команду меню File > Auto Save.

При написании и редактировании содержимого фалов используйте форматирование содержимого файла командой Format Document с помощью горячих клавиш SHIFT + ALT + F.

Подробнее о базовых действиях и работе в редакторе VS Code:

- https://code.visualstudio.com/docs/editor/codebasics
- https://code.visualstudio.com/docs/languages/html

Расширения (Extensions)

Существует множество подключаемых модулей и расширений, которые часто называют плагины (plugins), позволяющих улучшить процесс разработки веб-разработки.

Рассмотрим некоторые из них.

Расширение Prettier - Code formatter и HTML Format – это плагины, которые автоматически форматирует код HTML, CSS и JavaScript. Это помогает поддерживать внешний вид кода аккуратным и организованным, упрощая его чтение и поддержку.

Расширение Live Server – это веб-сервер, который позволяет просматривать код HTML, CSS и JavaScript в браузере во время работы с ним. С помощью этого плагина можно видеть изменения в режиме реального времени и эффективно отлаживать код.

1.2. GitHub Pages

Сервер GitHub – позволяет загружать репозитории кода для хранения в системе управления версиями Git. Сервер GitHub предоставляет бесплатный сервис GitHub Pages, который позволяет публиковать веб-сайт. В результате размещенный на GitHub Pages веб-сайт доступен в сети Интернет.

В репозиториях на GitHub можно хранить любой код, но для полноценного использования функции GitHub Pages необходимо создать репозиторий <username>.github.io, где <username> – это ваше имя пользователя на сервере github.com. Код репозитория <username>.github.io должен быть структурирован как типичный веб-сайт, то есть, основной точкой входа должен быть HTML-файл с именем index.html.

В результате итоговый сайт будет доступен по адресу http(s)://<username>.github.io

Подробнее о сервисе GitHub Pages: https://docs.github.com/en/pages/getting-started-with-github-pages/creating-a-github-pages-site

2. Основы HTML, CSS и jQuery

2.1. Основы HTML

Веб-страницы отображаемые в веб-браузерах представляют собой результат визуализации (рендеринга) html-документа. Для описания html-документа используется язык гипертекстовой разметки HTML (HyperText Markup Language).

HTML состоит из ряда элементов, которые используются, чтобы вкладывать или оборачивать различные части контента, чтобы заставить контент отображаться или действовать определенным образом в окне веб-браузера.

HTML элемент определяется начальным тегом, некоторым содержимым и конечным тегом: <tagname> Контент находится здесь... </tagname>. HTML элемент это все от начального тега до конечного тега, например, <h1>Мой первый заголовок</h1> или Мой первый параграф .

Отдельные элементы объединяются в HTML страницу.

Элемент <html>

Элемент <html> является контейнером, который заключает в себе все содержимое веб-страницы, включая элементы <head> и <body>. Как правило, <html> идет в документе вторым, после определения типа документа (Document Type Definition, DTD), устанавливаемого через <!DOCTYPE>. Закрывающий тег </html> всегда стоит в документе последним.

Листинг 2.1. Пример структуры пустого html-документа

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

<!-- Этот раздел предназначен для заголовка страницы и технической информации. -->
</head>
<body>
<!-- Этот раздел предназначен для информативной части, которая будет отобраться в окне браузера. -->
</body>
</html>

Атрибуты веб-элементов

Элементы также могут иметь атрибуты (attribute): <tagname attribute> Контент находится здесь... </tagname>. Атрибуты содержат дополнительную информацию об элементе, которая используется при рендеринге (визуализации) веб-элемента в окне браузера.

Атрибут (attribute) всегда должен иметь следующий вид: name=″value″,

где

name – имя атрибута, за которым следует знак равенства.

value – значение атрибута, заключенное с двух сторон в кавычки.

Разметка текста

Рассмотрим основные HTML элементы, которые используются для разметки текста: абзац, заголовок, список.

Абзац

Тег определяет текстовый абзац (от англ. paragraph – абзац, параграф) и является блочным элементом, всегда начинается с новой строки.

Листинг 2.2. Пример оформления абзаца

Данный текст представляет собой абзац. Абзац начинается с новой строки. Абзац представляет собой блочный элемент

Листинг 2.3. Пример использования атрибута с именем align и значением center для выравнивания строчек абзаца по центу окна веб-браузера

Текст этого абзаца выровнен по центру страницы

Атрибут align

Атрибут align используется для выравнивания содержимого блока по его краям. Может быть применен для тега , <h> или любого блочного тега. Например, для абзаца текст может выравниваться внутри блока по левому краю, по правому краю или по центру блока.

Атрибут align="left" используется для выравнивания текста по левому краю. В этом случае строки текста выравнивается по левому краю, а правый край располагается «лесенкой». Такой способ выравнивания является наиболее популярным на сайтах, поскольку позволяет пользователю легко отыскивать взглядом новую строку и комфортно читать большой текст.

Атрибут align="center" позволяет выровнять текст по центру. Текст помещается по центру горизонтали окна браузера или контейнера, где расположен текстовый блок. Подобный способ выравнивания активно используется в заголовках и различных подписях, вроде подрисуночных, он придает официальный и солидный вид оформлению текста.

Атрибут align="justify" представляет выравнивание по ширине – это означает одновременное выравнивание по левому и правому краю. Чтобы произвести это действие браузер в этом случае добавляет пробелы между словами.

Заголовок

HTML предлагает шесть заголовков разного уровня, которые показывают относительную важность секции, расположенной после заголовка. Так, тег <h1> представляет собой наиболее важный заголовок первого уровня, а тег <h6> служит для обозначения заголовка шестого уровня и является наименее значительным. По умолчанию, заголовок первого уровня отображается самым крупным шрифтом жирного начертания, заголовки последующего уровня по размеру меньше. Теги <h1>,...,<h6> относятся к блочным элементам, они всегда начинаются с новой строки, а после них другие элементы отображаются на следующей строке. Кроме того, перед заголовком и после него добавляется пустое пространство.

Листинг 2.4. Пример заголовков

```
<h1>Это заголовок первого уровня – самый важный</h1>
<h2>Это заголовок второго уровня. Текст отображается крупным шрифтом жирного начертания. </h2>
<h6>Заголовок шестого уровня и является наименее значительным.</h6>
```

Списки

Для отображения в браузере списков HTML имеет специальные элементы для них. Разметка списка всегда как минимум из двух элементов. Наиболее распространенными типами списков являются нумерованные и ненумерованные списки.

Ненумерованные списки (unordered list) – это списки, где порядок пунктов не имеет значения, как в списке покупок. Они оборачиваются в элемент <ul>.

Нумерованные списки (ordered list) – это списки, где порядок пунктов имеет значение, как в рецепте. Они оборачиваются в элемент <ol>.

Каждый пункт внутри списков располагается внутри элемента <li> (list item, элемент списка).

Листинг 2.5. Пример ненумерованного списка

```
<ul>
  <li>Это первый элемент ненумерованного списка.</li>
  <li>Второй элемент.</li>
  <li>Последний элемент ненумерованного списка.</li>
</ul>
```

Маркеры элементов ненумерованного списка могут принимать один из трех видов: круг (по умолчанию), окружность и квадрат. Для выбора стиля маркера используется атрибут type тега <ul>:

- type="disc" – список с маркерами в виде круга;
- type="circle" – список с маркерами в виде окружности;
- type="square" – список с квадратными маркерами.

В качестве нумерующих элементов могут выступать следующие значения:

- арабские числа (1, 2, 3, ...);
- прописные латинские буквы (A, B, C, ...);
- строчные латинские буквы (a, b, c, ...);
- прописные римские числа (I, II, III, ...);

- строчные римские числа (i, ii, iii, ...).

Для указания типа нумерованного списка применяется атрибут type тега <ol>:

- type="1" – арабские числа;
- type="A" – прописные буквы латинского алфавита
- type="a" – строчные буквы латинского алфавита
- type="I" – римские числа в верхнем регистре
- type="i" – римские числа в нижнем регистре.

Чтобы начать список с определенного значения, используется атрибут start тега <ol>. При этом не имеет значения, какой тип списка установлен с помощью type, атрибут start одинаково работает и с римскими и с арабскими числами.

Гипертекстовые ссылки

Элемент <a> (от англ. anchor – якорь) является одним из важных в HTML и предназначен для создания ссылок. Для этого необходимо сообщить браузеру, что является ссылкой, а также указать адрес документа, на который следует сделать ссылку. Самый важный атрибут элемента <a> – это атрибут href, что указывает на пункт назначения ссылки. Текст ссылки это та часть, которая будет видна читателю. Нажав на текст ссылки, читатель отправится на указанный URL-адрес.

Листинг 2.6. Пример гипертекстовый ссылки.

```
<a href=”https://www.polessu.by/”>Этот текст является ссылкой на сайт университета</a>
```

В качестве значения атрибута href (англ. hypertext reference – гипертекстовая ссылка) используется адрес документа, на который происходит переход. Адрес ссылки может быть абсолютным и относительным. Абсолютные адреса работают везде и всюду независимо от имени сайта или веб-страницы, где прописана ссылка. Относительные ссылки, как следует из их названия, построены относительно текущего документа или корня сайта.

Якорем называется закладка с уникальным именем на определенном месте веб-страницы, предназначенная для создания перехода к ней по ссылке. Якоря удобно применять в документах большого объема, чтобы можно было быстро переходить к нужному разделу.

Для создания якоря следует вначале сделать закладку в соответствующем месте и дать ей имя при помощи атрибута name или id тега <a>. В качестве значения href для перехода к этому якорю используется имя закладки с символом решетки (#) впереди.

Листинг 2.7. Пример якоря и ссылки на него

```
<p><a name="top"></a></p>
...
<p><a href="#top">Наверх</a></p>
```

Между тегами и текст не обязателен, так как требуется лишь указать местоположение перехода по ссылке, находящейся внизу страницы. Имя ссылки на якорь начинается с символа #, после чего идет имя якоря, оно выбирается любое, соответствующее тематике. Главное, чтобы значения атрибутов name и href совпадали без учета символа решетки.

Изображения

Элемент <img> (англ. image – изображение) предназначен для отображения на веб-странице изображений в графическом формате GIF, JPEG, SVG или PNG. Адрес файла с

картинкой задается через атрибут src. Если необходимо, то рисунок можно сделать ссылкой на другой файл, поместив <img> в контейнер <a>.

Листинг 2.8. Пример вставки изображения.

<img src=" https://www.polessu.by/sites/default/files/images/26-shestakova/scale_1200.png">

Элемент <img> относится к элементам, которые не имеют контента, и называются пустыми элементами. Как видно из Листинга 2.8. он не имеет закрывающего тега </img>, и никакого внутреннего контента. Это потому, что элемент изображения не оборачивает контент для влияния на него. Его целью является вставка изображения в HTML страницу в нужном месте.

Изображение может улучшить дизайн и внешний вид веб-страницы. Часто используемые атрибуты для тега <img> являются: alt, width, border.

Атрибут alt

Атрибут alt – альтернативный текст для изображения. Альтернативный текст – это «пояснительный текст». Он должен предоставить читателю достаточно информации, чтобы иметь представление о том, что передает изображение. Всегда давайте подходящий альтернативный текст для вашего изображения!

Атрибут width

Атрибут width – ширина изображения. Для изменения размеров изображения средствами HTML предусмотрены атрибуты height и width. Допускается использовать значения в пикселах или процентах. Если установлена процентная запись, то размеры изображения вычисляются относительно родительского элемента – контейнера, где находится тег <img>. В случае отсутствия родительского контейнера, в его качестве выступает окно браузера. Иными словами, width="100%" означает, что рисунок будет растянут на всю ширину веб-страницы. Добавление только одного атрибута width или height сохраняет пропорции и отношение сторон изображения. Браузер при этом ожидает полной загрузки рисунка, чтобы определить его первоначальную высоту и ширину.

Обязательно задавайте размеры всех изображений на веб-странице. Это несколько ускоряет загрузку страницы, поскольку браузеру нет нужды вычислять размер каждого рисунка после его получения.

Ширину и высоту изображения можно менять как в меньшую, так и большую сторону. Однако на скорость загрузки рисунка это никак не влияет, поскольку размер файла остается неизменным. Поэтому с осторожностью уменьшайте изображение, т.к. это может вызвать недоумение у читателей, отчего такой маленький рисунок так долго грузится. А вот увеличение размеров приводит к обратному эффекту – размер изображения велик, но файл относительно изображения аналогичного размера загружается быстрее. Но качество рисунка при этом ухудшается.

Атрибут border

Атрибут border – толщина рамки вокруг изображения. Изображение, помещаемое на веб-страницу, можно поместить в рамку различной ширины. Для этого служит атрибут border тега <img>. По умолчанию рамка вокруг изображения не отображается за исключением случая, когда рисунок является ссылкой. Чтобы убрать рамку, следует задать атрибут border="0".

Лабораторное занятие № 1

Одностраничный сайт с блочной html-верстkой. Сверстайте html-документ содержащий заголовок, меню в виде маркированного списка. Элементы меню являются ссылками на разделы html-документа. Разделы оформить в виде заголовков, текстов-параграфов и изображения. В качестве источников изображений использовать изображения из Интернета.

Примеры html-документов:

- landing page сайта-витрины организации, предпринимателя, профессиональной деятельности;

- html-документ ПолесГУ, меню и разделы: Университет, Библиотека, Образование.

**Блок **

Блок представляет собой базовый блочный элемент и используется для разделения контента HTML. Блок не влияет на контент или макет до тех пор, пока не будет стилизован с помощью CSS.

Листинг 2.9. Пример использования блока

```
<div>
  <p>Любой тип контента. Например, &lt;p&gt;, &lt;table&gt;. Все что угодно!</p>
</div>
```

Результат отображения в браузере представленного выше фрагмента html-документа с использованием блока будет такой же как и без использования блока . Но как только блочный элемент будет стилизован с использованием атрибута class, то мы сможем увидеть результат форматирования контента в окне браузера.

Листинг 2.10. Пример стилизованного блока .

```
<div class="gradientbox">
  <p>Вот очень интересная заметка в прекрасном прямоугольнике с градиентной заливкой фона и рамкой.</p>
</div>
```

Листинг 2.11. Пример стилевого оформления (CSS) использованного в Листинге 2.10.

```
.shadowbox {
  width: 15em;
  border: 1px solid #333;
  padding: 8px 12px;
  background-image: linear-gradient(180deg, #fff, #ddd 40%, #ccc);
}
```

Современные рекомендации по использованию блочных элементов гласят, что элемент следует использовать только в том случае, если никакой другой семантический элемент (такой как <article> или <nav>) не подходит.

Семантические секционные элементы

Семантический элемент четко описывает его значение как для браузера, так и для разработчика. Стандарт HTML5 предлагает следующие семантические элементы для определения различных частей веб-страницы: <article>, <aside>, <details>, <figcaption>, <figure>, <footer>, <header>, <main>, <mark>, <nav>, <section>, <summary>, <time>.

Рассмотрим часто используемые семантические элементы: <section>, <article>, <nav>, <header>, <footer>.

Элемент секции <section>

Веб-элемент <section> представляет собой автономный раздел – который не может быть представлен более точным по семантике элементом – внутри HTML-документа. Как правило, но не всегда, разделы имеют заголовок.

Например, меню навигации должно быть помещено в элемент <nav>, но список результатов поиска и отображение карты с ее элементами управления не имеют специфических элементов и могут быть помещены в <section>.

Каждый элемент <section> должен быть идентифицирован, обычно путем добавления заголовка (элементы <h1>) в качестве дочернего элемента.

Если имеет смысл по-особому объединить содержимое элемента <section> (например, сделать цельным и независимым разделом HTML-документа), используйте вместо него элемент <article>.

Не используйте элемент <section> как общий контейнер; для этого есть , особенно когда секционирование применяется только для стилизации. На практике раздел должен логически выделяться в структуре документа.

Фрагмент содержимого <article>

HTML-элемент <article> представляет самостоятельную часть документа, страницы, приложения или сайта, предназначенную для независимого распространения или повторного использования. Этот элемент может представлять статью на форуме, статью в журнале или газете, запись в блоге или какой-либо другой самостоятельный фрагмент содержимого.

Данный документ может иметь множество статей; например, когда читатель просматривает блог, в котором текст каждой статьи отображается один за другим, каждая публикация будет находиться в элементе <article>, возможно, с одним или более элементами <section> внутри.

Каждый элемент <article> должен быть идентифицирован, обычно путем добавления заголовка (элементы <h1>-<h6>) в качестве дочернего элемента.

Когда элемент <article> является вложенным, внутренний элемент представляет собой контент связанный с внешним элементом. Например, комментарии к публикации в блоге могут быть элементами <article>, вложенными в другой <article>, являющийся публикацией в блоге.

Информация об авторе в элементе <article> может быть представлена через элемент <address>, но это не применимо к вложенным элементам <article>.

Дата и время публикации в элементе <article> могут быть описаны с помощью атрибута datetime элемента <time>.

Элемент навигации <nav>

HTML-элемент <nav> определяет отдельную секцию документа, назначение которой обозначение ссылок навигации (как внутри текущего документа, так и ведущих на другую страницу). В качестве примера такой секции можно привести меню, якорные ссылки.

Не обязательно все ссылки должны быть обернуты в <nav>. <nav> следует использовать лишь для главных навигационных блоков. Например, <footer> часто содержит список ссылок, которые не стоит оборачивать в <nav> .

Документ может содержать несколько <nav> элементов. Например, один для навигации по сайту, второй для навигации по странице.

Заголовочный элемент <header>

HTML-элемент <header> представляет собой вводный контент, обычно группу вводных или навигационных средств. Он может содержать другие элементы-заголовки, а также логотип, форму поиска, имя автора и другие элементы.

Элемент <header> не относится к секционному контенту , а значит не создает новый раздел в структуре HTML-документа. При этом элемент <header> обычно должен содержать заголовок ближайшего раздела (элементы h1-h6), но это не обязательно.

Нижний колонтитул <footer>

HTML-элемент <footer> представляет собой нижний колонтитул (футер, подвал) для своего ближайшего секционного контента или секционного корня. Футер обычно содержит информацию об авторе раздела, информацию об авторском праве или ссылки на связанные документы.

Заключите информацию об авторе в элемент <address>, который может быть добавлен в элемент <footer>.

Элемент <footer> не относится к секционному контенту, а значит не создает новый раздел в структуре HTML-документа.

Атрибут style

Атрибут style применяется для определения стиля элемента с помощью правил CSS. Синтаксис использования следующий: style="стилевые правила". В качестве стилевых правил используются записи в виде: вначале следует имя стилевого свойства, затем через двоеточие его значение. Стилевые свойства разделяются между собой точкой с запятой: style="имя_стилевого_свойства_1: значение_1; имя_стилевого_свойства_2: значение_2".

Пример использования атрибута style для изменения цвета текста внутри текущего абзаца представлен ниже.

Листинг 2.11. Пример использования style для изменения цвета текста абзаца.

```
<p style ="color:blue;">
  Это параграф синего цвета.
</p>
```

В настоящее время использование атрибута style для оформления элементов не рекомендуется. Это связано в первую очередь с тем, что в настоящее время рекомендуется использовать HTML для вывода информации, а оформление выполнять с помощью CSS. Вынесение стилевого описания в отдельный файл позволяет упростить последующие правки сайта, которые проще делать в CSS, а не искать нужный код в файлах и заменять атрибуты style.

Таким образом, указание стилей в атрибуте style имеет свои недостатки: дублирование кода, сложность изменения, если стиль одинаковый для разных элементов, высокий приоритет правил – не переопределяешь через стили для всей страницы. В небольших проектах использование атрибута style может быть совершенно достаточно.

Тег <style>

HTML-элемент <style> содержит стилевую информацию для документа или его части. По умолчанию стилевые инструкции внутри этого элемента считаются написанными на CSS. Тег <style> применяется для определения стилей элементов веб-страницы. Тег <style> необходимо использовать внутри контейнера <head>. Можно задавать более чем один тег <style>.

Листинг 2.12. Синтаксис использования тега <style>.

```
<head>
  <style type="text/css">
   стилевые правила
  </style>
</head>
```

Атрибут type сообщает браузеру, какой синтаксис использовать, чтобы правильно интерпретировать стили. Так же может использоваться атрибут media который определяет устройство вывода, для работы с которым предназначена таблица стилей.

Листинг 2.13. Пример использования тега <style> для описания стилевого правила цвета текста абзаца

```
<head>
  <style type="text/css">
    p {
        color: green;
      }
  </style>
</head>
```

Тег <link>

Описание стилевых правил размещенных в теге <style> внутри тега <head> можно вынести в отдельный файл. Для этого необходимо создать текстовый документ представляющий собой файл таблицы стилей, имеющий расширение .css. В качестве имени файла таблицы стилей используется имя style.css и файл обычно размещается в корневой папке вместе с html-документами или в отдельной папке styles.

Структура хранения файлов веб сайта

\root (корневая папка проекта)

\ index.html (html-страница)

\ style.css (css-файл таблицы стилей)

Чтобы связать styles.css с index.html, необходимо добавить следующую строку внутри тега <head> HTML документа:

Листинг 2.14. Связывание внешней таблицы стилей с html страницей.

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

Таким образом, внешняя таблица стилей - это когда есть отдельный CSS файл с расширением .css, и ссылка на него выполняется из HTML-элемента с помощью тега <link>.

Это самый распространенный и полезный способ крепления CSS к документу, так в данном случае можно привязать CSS сразу к нескольким страницам, что позволяет стилизовать их все с той же таблицей стилей. В большинстве случаев различные страницы сайта будут выглядеть почти так же, поэтому можно использовать один и тот же набор правил для основного вида.

2.2. Основы CSS

CSS (Cascading Style Sheets) – это код, который используется для стилизации веб-страницы. Как и HTML, CSS на самом деле не является языком программирования. Это не язык разметки – это язык таблицы стилей. Это означает, что он позволяет применять стили выборочно к элементам в документах HTML. Например, чтобы выбрать все

элементы абзаца <p> на HTML странице и установить \ задать \ форматировать цвет текст внутри них на красный, необходимо написать CSS.

Листинг 2.15. Пример CSS для форматирования цвета текста абзаца.

```
p {
  color: red;
}
```

Селекторы – это шаблоны, которые используются для привязки стилевых свойств к элементам в документе. Можно задать стиль для всех элементов или сократить выбор с помощью определенного селектора. Например, селекторы типа выбирают элементы HTML-документа по их тегу. Например, селектор p выберет все <p> на странице.

Указанный в Листинге 2.15 пример CSS для форматирования цвета абзаца называется CSS-правилом.

Правила CSS

Рассмотрим составные части правила.

Селектор (Selector) – имя HTML-элемента в начале набора правил. Он выбирает элементы для применения стиля. В Листинге 2.15 это элементы p абзацы <p>.

Объявление (Declaration) – представляет собой правило, например, color: red;. В данном правиле указывается свойство (Property) color элемента <p> которое будет форматировано значением (Value) red.

Важные части синтаксиса:

- Каждый набор правил (кроме селектора) должен быть обернут в фигурные скобки {}.

- В каждом объявлении необходимо использовать двоеточие : , чтобы отделить свойство от его значений.

- В каждом наборе правил должна использоваться точка с запятой ; , чтобы отделить каждое объявление от следующего.

Таким образом, чтобы изменить несколько значений свойств одновременно, эти свойства нужно перечислить, разделяя их между собой точкой с запятой.

Листинг 2.16. Пример CSS для задания цвета текста абзаца, ширины контейнера содержащего текст абзаца и его границы.

```
p {
  color: red;
  width: 500px;
  border: 1px solid black;
}
```

CSS селекторы

Селектор – определяет, к какому элементу применять то или иное CSS-правило. Универсальный селектор выбирает все элементы. По желанию, он может быть ограничен определенным пространством имен или относиться ко всему пространству имен. Пример: * будет соответствовать всем элементам на странице.

Селекторы по типу элемента. Этот базовый селектор выбирает тип элементов, к которым будет применяться правило. Пример: селектор input выберет все элементы <input>.

Селекторы по классу – этот селектор выбирает элементы, основываясь на значении их атрибута class. Пример: селектор .index выберет все элементы с соответствующим классом, который был определен в атрибуте class="index".

Селекторы по идентификатору. Этот базовый селектор выбирает элементы, основываясь на значении их id атрибута. Не забывайте, что идентификатор должен быть уникальным, т. е. использоваться только для одного элемента в HTML-документе. Пример: селектор #toc выберет элемент с идентификатором toc, который был определен в атрибуте id="toc".

Комбинатор запятая «,» это способ группировки, он выбирает все совпадающие узлы. Пример: div, span выберет оба элемента: и .

Псевдоклассы – знак «:» позволяет выбрать элементы, основываясь на информации, которой нет в дереве элементов.

Например, для тега <a> существуют следующие псевдоклассы:

- a:hover соответствует всем элементам <a> которые имеют статус «посещенные»;
- a:hover соответствует элементу, над которым проходит указатель мыши;
- input:focus соответствует полю ввода, которое получило фокус.

Дополнительные материалы:

- https://developer.mozilla.org/ru/docs/Web/CSS/Pseudo-classes

CSS свойства

CSS позволяет тегам HTML назначать любые визуальные стили, формируя уникальный внешний вид сайта. Делается это при помощи свойств CSS. Рассмотрим наиболее часто используемые свойства CSS с описанием, допустимыми значениями и примерами.

Свойство font-family задает семейство шрифта, которое будет использоваться для оформления текста. Если имя шрифта состоит из более одного слова, например, Trebuchet MS, то оно должно заключаться в кавычки. Список шрифтов может включать одно или несколько названий, разделенных запятой. Браузер будет использовать первый, найденный на компьютере пользователя, шрифт из заданного списка.

Когда браузер встречает первый шрифт из списка, то он проверяет его наличие на компьютере пользователя. Если такого шрифта нет, проверяется следующее имя из списка.

Таким образом, указание нескольких шрифтов увеличивает вероятность, что хотя бы один из них будет обнаружен на компьютере пользователя.

В конце списка всегда следует указывать ключевое слово, которое описывает тип шрифта, например, serif, sans-serif, cursive, fantasy или monospace.

Листинг 2.17. Пример использования списка шрифтов для форматирования параграфов.

```
p {
  font-family: "Times New Roman", Georgia, Serif;
}
```

Свойство font-size – определяет размер шрифта.

Листинг 2.18. Пример использования установки размера шрифта для заголовков и параграфов.

```
h1 {
  font-size: 200%;
```

```
}
p {
  font-size: 15px;
}
```

Свойство list-style-image устанавливает изображение, которое будет использоваться в качестве маркера элементов списка.

Листинг 2.19. Пример использования установки изображения для маркеров элементов списка.

```
ul {
  list-style-image: url('sqpurple.gif');
}
```

Дополнительно всегда определяйте свойство list-style-type. Значение этого свойства будет использоваться, если изображение по какой-либо причине будет недоступно. Свойство list-style-type задает тип маркеров для использования в списке, например, квадратные или круглые маркеры для неупорядоченного списка, или цифры, буквы или римские цифры для упорядоченного списка.

Свойство list-style-type устанавливает вид маркера элементов списка. Значения зависят от того, к какому типу списка они применяются: маркированному или нумерованному.

Для маркированного списка свойство list-style-type может принимать следующие значения:

- circle – маркер в виде кружка;
- disc – маркер в виде точки;
- square – маркер в виде квадрата.

Для нумерованных списков значение свойства list-style-type определяет стиль маркеров:

- decimal – арабские числа (1, 2, 3, 4,...);
- lower-alpha – строчные латинские буквы (a, b, c, d,...);
- upper-alpha – заглавные латинские буквы (A, B, C, D,...);
- lower-roman – римские числа в нижнем регистре (i, ii, iii, iv, v,...);
- upper-roman – римские числа в верхнем регистре (I, II, III, IV, V,...)
- none – отменяет маркеры для списка.

Листинг 2.20. Пример определения нескольких различных стилей списков.

```
ul.circle {
  list-style-type: circle;
}
ul.square {
  list-style-type: square;
}
ol.upper-roman {
  list-style-type: upper-roman;
}
ol.lower-alpha {
  list-style-type: lower-alpha;
}
```

Свойство list-style-position определяет, как будет размещаться маркер относительно текста.

Имеется два значения:

- inside – маркер является частью текстового блока и отображается в элементе списка;

- outside – текст выравнивается по левому краю, а маркеры размещаются за пределами текстового блока.

Листинг 2.21. Пример установки положения маркера для элементов списка когда маркер является частью текстового блока

```
ul {
  list-style-position: inside;
}
```

Свойство color – это свойство определяет цвет текста элемента. CSS позволяет использовать почти 16.777.216 цветов. Они могут быть представлены либо предопределенным именем, либо функциональным значением RGB, RGBA, HSL, HSLA, либо шестнадцатеричным кодом (HEX).

Листинг 2.22. Пример установки цвета текста с предопределенным именем Tomato, его шестнадцатеричным кодом и его функциональными значениями RGB и HSL.

```
h1 {
  color: tomato;
}
h2 {
  color: #ff6347;
}
h3 {
  color: rgb(255, 99, 71);
}
h4 {
  color: hsl(9, 100%, 64%);
}
```

Свойство background-color определяет фоновый цвет элемента. Фоном элемента считается пространство общим размером самого элемента плюс его поля (свойство padding) и рамка (свойство border), но не отступы (свойство margin). Цвет может быть представлен предопределенным именем или функциональным значением RGB, RGBA, HSL, HSLA, а так же шестнадцатеричным кодом (HEX).

Листинг 2.23. Пример установки фонового цвета текста для заголовка.

```
h1 {
  background-color: #00ff00;
}
```

Свойство height устанавливает высоту блочных или заменяемых элементов (к ним относится тег <img>). Высота не включает толщину границ вокруг элемента, значение отступов и полей.

Если содержимое блока превышает указанную высоту, то высота элемента останется неизменной, а содержимое будет отображаться поверх него. Из-за этой особенности может получиться наложение содержимого элементов друг на друга, когда

элементы в коде HTML идут последовательно. Чтобы этого не произошло, следует добавить overflow: auto к стилю элемента.

Свойство width устанавливает ширину блочных или заменяемых элементов. Устанавливает ширину блочных или заменяемых элементов (к ним относится тег <img>). Ширина не включает толщину границ вокруг элемента, значение отступов и полей.

Листинг 2.24. Пример установки высоты и ширины параграфа.

```
p {
  height: 100px;
  width: 100px;
}
```

Свойство border-style – это свойство определяет стиль всей рамки вокруг элемента. Это свойство определяет стиль всех четырех сторон рамки HTML элемента.

Свойство border-style может принимать от одного до четырех значений:

- border-style: dotted; – все четыре стороны рамки состоят из точек (dotted);

- border-style: dotted solid; – верхняя и нижняя стороны рамки состоят из точек (dotted), правая и левая стороны рамки сплошная линия (solid);

- border-style: dotted solid double; – верхняя сторона рамки состоит из точек (dotted), правая и левая стороны рамки сплошная линия (solid), нижняя сторона рамки двойная сплошная линия (double).

- border-style: dotted solid double dashed; – верхняя сторона рамки состоит из точек (dotted), правая сторона рамки сплошная линия (solid), нижняя сторона рамки двойная сплошная линия (double), левая сторона рамки состоит из черточек (dashed).

Возможные значения:

- none – значение по умолчанию, без рамки;

- solid – сплошная линия;

- double – двойная сплошная линия;

- dotted – рамка состоит из точек;

- dashed – рамка состоит из черточек.

Листинг 2.25. Пример установки стиля для четырех сторон рамки параграфа.

```
p {
  border-style: solid;
}
```

Свойство border-width – это свойство определяет толщину всех четырех сторон рамки HTML элемента.

Свойство border-width может принимать от одного до четырех значений:

- border-width: 1px; – все четыре стороны рамки 1px;

- border-width: 1px 2px; – верхняя и нижняя стороны рамки 1px, правая и левая стороны рамки 2px;

- border-width: 1px 2px 3px; – верхняя сторона рамки 1px, правая и левая стороны рамки 2px, нижняя сторона рамки 3px;

- border-width: 1px 2px 3px 4px; – верхняя сторона рамки 1px, правая сторона рамки 2px, нижняя сторона рамки 3px, левая сторона рамки 4px.

Следует всегда определять свойство border-style перед свойством border-width. Элемент должен иметь рамку, прежде чем менять ее толщину. Толщина рамки может быть задана с помощью предопределенного имени: thin, medium, thick.

Листинг 2.26. Пример установки толщины 15px для всех четырех сторон рамки параграфа.

```
p {
  border-style: solid;
  border-width: 15px;
}
```

Свойство border-color – это свойство определяет цвет всех четырех сторон рамки HTML элемента.

Свойство border-color может принимать от одного до четырех значений:

- border-color: red; – все четыре стороны рамки красные (red);

- border-color: red green; – верхняя и нижняя стороны рамки красные (red), правая и левая стороны рамки зеленые (green);

- border-color: red green blue; – верхняя сторона рамки красная (red), правая и левая стороны рамки зеленые (green), нижняя сторона рамки синяя (blue);

- border-color: red green blue pink; – верхняя сторона рамки красная (red), правая сторона рамки зеленая (green), нижняя сторона рамки синяя (blue), левая сторона рамки розовая (pink).

Следует всегда определять свойство border-style перед свойством border-color. Элемент должен иметь рамку, прежде чем менять ее цвет.

Листинг 2.27. Пример установки цвета для верхней и нижней сторон #ff0000, и для левой и правой сторон #0000ff рамки параграфа.

```
p {
  border-style: solid;
  border-color: #ff0000 #0000ff;
}
```

Свойство border-radius устанавливает радиус скругления уголков рамки. Если рамка не задана, то скругление также происходит и с фоном. Свойство border-radius является универсальным свойством, которое позволяет за одну декларацию определить свойства border-top-left-radius, border-top-right-radius, border-bottom-right-radius и border-bottom-left-radius. Если задается только одно значение, то оно будет определять радиус для всех 4 углов рамки элемента.

Для того чтобы установить радиус для каждого угла отдельно необходимо следовать следующим правилам:

- четыре значения: первое значение – верхний левый угол, второе – верхний правый угол, третье – нижний правый угол и четвертое значение – нижний левый угол;

- три значения: первое значение – верхний левый угол, второе значение – верхний правый и нижний левый углы и третье значение – нижний правый угол;

- два значения: первое значение – верхний левый и нижний правый углы, второе значение – верхний правый и нижний левый углы;

- одно значение: у всех четырех углов одинаковый радиус скругления.

Листинг 2.28. Пример установки скругленных углов рамки для элемента .

```
div {
  border: 2px solid;
  border-radius: 25px;
}
```

В случае задания двух параметров для скругления через слэш «/», то первый задает радиус по горизонтали, а второй по вертикали (эллиптические уголки).

Отступом является расстояние от внешнего края границы текущего элемента до внутренней границы его родительского элемента.

Для задания отступов используются следующие свойства:

- margin-bottom – это свойство устанавливает размер нижнего отступа элемента;
- margin-left – это свойство устанавливает размер левого отступа элемента;
- margin-right – это свойство устанавливает размер правого отступа элемента;
- margin-top – это свойство устанавливает размер верхнего отступа элемента.

Полем называется расстояние от внутреннего края рамки элемента до воображаемого прямоугольника, ограничивающего его содержимое.

Для задания полей используются следующие свойства:

- padding-bottom – это свойство устанавливает размер нижнего поля элемента;
- padding-left – это свойство устанавливает размер левого поля элемента;
- padding-right – это свойство устанавливает размер правого поля элемента;
- padding-top – это свойство устанавливает размер верхнего поля элемента.

Для отступов и полей допускается использование отрицательных значений.

Возможные значения свойств для отступов и полей:

- auto – браузер автоматически устанавливает размер отступа или поля;
- размер – устанавливает размер отступа или поля в единицах CSS: пикселях (px), сантиметрах (cm) и т.д. Значение размера по умолчанию 0.
- % – устанавливает размер отступа или поля в процентах от ширины родительского элемента.

Листинг 2.29. Пример установки размера нижнего поля 2cm и верхнего отступа 1px элемента <p>.

```
p {
  padding-bottom: 2cm;
  margin-top: 1px;
}
```

Единицы измерения

Указывать длину в CSS можно в разных единицах. Некоторые из них пришли из типографской традиции, как пункт (pt) и пика (pc), другие, напр. сантиметр (cm), миллиметры (mm) и дюйм (in).

Пиксель px – это самая базовая, абсолютная и окончательная единица измерения. Количество пикселей задается в настройках разрешения экрана, один px – это как раз один такой пиксель на экране. Все значения браузер в итоге пересчитает в пиксели.

Для отображения элементов на экранах электронных устройств World Wide Web Consortium (https://www.w3.org/Style/Examples/007/units.en.html) рекомендует использовать следующие единицы: px, em, %.

Единица em – просто размер шрифта. В элементе, которому задан шрифт в 24px, 1em и означает эти 24px. Указание размеров (например, для отступов) в em означает, что они задаются относительно шрифта, и какой бы ни был шрифт у пользователя – крупный (например, на большом экране) или мелкий (например, на мобильном устройстве), эти размеры останутся пропорциональными. Объявления наподобие margin: 1em в CSS крайне популярны.

Листинг 2.30. Пример использования единиц em.

```
<body style="font-size:20px">
  <p>Текст основного документа</p>
  <div style="font-size:2em">
     Текст основного блока для которого задан font-size:2em
     <div style="font-size:2em">Текст для которого задан font-size:2rem вложенного блока в основной блок</div>
  </div>
</body>
```

В Листинге 2.30 для тега <body> задан размер текста 20px. Этим размером будет выведен на экран фрагмент текста заключенный в тег Текст основного документа . Размер для фрагмента текста заключенного в первый тег будет рассчитан на основе размера шрифта установленного в теге <body> умноженного на два. В результате фраза «Текст основного блока…» будет выведена на экран с размером текста 40px. Второй блок вложен в первый блок и размер его фрагмента текста будет уже определяться размером текста для первого блока.

Таким образом, для вывода на экран фразы «Текст … вложенного блока в основной блок» будет использован размер текста 80px.

Один rem (от «root em», т.е. «корневой em» или «em корневого элемента») – это размер шрифта корневого элемента в документе. В отличие от em, который может быть для каждого элемента свой, rem для всего документа один и тот же. Единица rem задает размер относительно размера шрифта элемента <html>.

Листинг 2.31. Пример использования единиц rem.

```
<html style="font-size:20px">
  <body>
    <p>Текст основного документа</p>
    <div style="font-size:2rem">
      Текст основного блока для которого задан font-size:2rem
      <div style="font-size:2rem">Текст для которого задан font-size:2rem вложенного блока в основной блок</div>
    </div>
  </body>
</html>
```

В Листинге 2.31 размер шрифта в первом блоке и во вложенном в него втором блоке задан в единицах rem – это значит что оба фрагмента текста будут выведены на экран одинакового размера равного 40px.

Лабораторное занятие № 2

Применение каскадных таблиц стилей. На основе сайта-витрины реализовать использование каскадных таблиц стилей для визуального форматирования содержимого landing page.

Контрольная точка №1

Верстка веб-страницы средствами HTML и CSS – демонстрация веб-сайта интернет-витрины.

2.3. Основы jQuery

Библиотека jQuery включает в себя набор функций JavaScript, фокусирующийся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими.

Библиотека jQuery, как правило, включается в веб-страницу как один внешний JavaScript-файл.

Листинг 2.32. Пример подключения библиотеки jQuery.

```
<head>
  <!-- 1-й метод — с локального сервера -->
  <script src="jquery-3.5.1.min.js">
  </script>
  <!-- 2-й метод — с сервера jquery.com или другого CDN -->
  <script src="https://code.jquery.com/jquery-3.5.1.min.js">
  </script>
</head>
```

Вся работа с jQuery ведется с помощью функции $. Если на сайте применяются другие JavaScript библиотеки, где $ может использоваться для своих нужд, то можно использовать ее синоним – jQuery. Второй способ считается более правильным, а чтобы код не получался слишком громоздким, можно писать его следующим образом.

Листинг 2.33. Пример объявления jQuery функции.

```
jQuery(function($) {
  // здесь код скрипта, где в $ будет находиться объект, предоставляющий доступ к функциям jQuery
})
```

Алгоритм работы с jQuery можно описать двумя способами:

1. Получение jQuery-объекта с помощью функции $(). Например, передав в нее CSS-селектор, можно получить jQuery-объект всех элементов HTML, попадающих под критерий и далее работать с ними с помощью различных методов jQuery-объекта. В случае, если метод не должен возвращать какого-либо значения, он возвращает ссылку на jQuery объект, что позволяет вести цепочку вызовов методов согласно концепции текучего интерфейса.

2. Вызов глобальных методов у объекта $, например, удобных итераторов по массиву.

Дополнительные материалы:

- Официальный сайт jQuery – https://jquery.com/.

- jQuery для начинающих – https://habr.com/ru/articles/38208/.

Рассмотрим, как с помощью библиотеки jQuery можно создать кликабельные блочные элементы на странице и галерею изображений.

Кликабельные блоки

Рассмотрим пример как сделать кликабельным блок с текстом, а не только ссылку. Для реализации будет использован список <ul> с классом class=«pane-list» в котором каждый элементы <li> будет кликабельными.

Листинг 2.33. Пример веб-элемента <li>.

```
<ul class="pane-list">
  <li>
    <h3><a href="https://www.polessu.by/">Сайт ПолесГУ</a></h3>
    <p>Полесский государственный университет</p>
  </li>
  ...
</ul>
```

Далее необходимо объявить JS-функцию.

Листинг 2.34. Пример JS-функции.

```
$(function () {
  $(".pane-list li").click(function () {
    window.location = $(this).find("a").attr("href");
    return false;
  });
});
```

Данная функция отслеживает событие click на веб-элементе <li> являющего частью списка <ul class="pane-list">. Когда пользователь кликает по элементу списка, функция произведет поиск тэга <a>, расположено в теге <li> элемента списка, на котором был выполнен клик и выполнит переход на страницу указанную в атрибуте href.

Для работы скрипта необходимо подключить JQuery библиотеку.

Листинг 2.35. Пример подключения библиотеки jQuery.

```
<head>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.4.1/jquery.js"
integrity="sha256-WpOohJOqMqqyKL9FccASB9O0KwACQJpFTUBLTYOVvVU="
crossorigin="anonymous">
  </script>
</head>
```

Демонстрацию примера можно увидеть на странице https://anton.shevchuk.name/wp-demo/jquery-tutorials/block-clickable.html.

Галерея изображений

Рассмотрим возможность реализовать галерею изображений, без перезагрузки веб-страницы. Данный подход предполагает предварительную загрузку всех миниатюр изображений галереи вместе с загрузкой контента веб-страницы.

Галерея представляет собой следующую структуру.

Листинг 2.36. Пример структуры веб-элементов галереи изображений.

```
<h2>Заголовок для галереи
  <em>Текстовое описание текущего, отображаемого изображения</em>
</h2>
<p><img id="largeImg" src="images/изображение-большой-вариант.jpg"
alt="Текстовое описание изображения"></p>
```

```
<p class="thumbs">
  <a href="images/изображение-большой-вариант.jpg " title=" Текстовое описание изображения"><img src="images/изображение-миниатюра.jpg"></a>
…
</p>
```

Веб-элемент <h2> содержит текст заголовка для всей галереи и веб-элемент отображающий текстовое описание изображения, которое в данный момент отображается. Элемент текстового абзаца <p> содержит тег <img id="largeImg"> для отображения выбранного изображения. Атрибут id="largeImg" используется в JS функции для указания места, куда загружать выбираемые в галерее изображения.

Элемент текстового абзаца <p class="thumbs"> содержит список гипертекстовых ссылок с миниатюрами изображений галереи и адресами больших изображений для последующей загрузки в тег <img id="largeImg">.

Далее необходимо объявить JS функцию.

Листинг 2.37. Пример JS функции.

```
<script type="text/javascript">
  $(function () {
    $("h2").append('<em></em>');
    $(".thumbs a").click(function () {
      var largePath = $(this).attr("href");
      var largeAlt = $(this).attr("title");
      $("#largeImg").attr({src: largePath, alt: largeAlt});
      $("h2 em").html(" (" + largeAlt + ")");
      return false;
    );
  });
</script>
```

Для работы скрипта необходимо подключить JQuery библиотеку.

Листинг 2.38. Пример подключения библиотеки jQuery.

```
<head>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.4.1/jquery.js"
integrity="sha256-WpOohJOqMqqyKL9FccASB9O0KwACQJpFTUBLTYOVvVU="
crossorigin="anonymous"></script>
</head>
```

Демонстрацию примера можно увидеть на странице https://anton.shevchuk.name/wp-demo/jquery-tutorials/img-replacement.html

Дополнительные материалы:

- Слайдеры и скроллеры – https://pcvector.net/scripts/slideshow_and_scroller
- Галереи изображений – https://sachinchoolur.github.io/lightslider/examples.html

Лабораторное занятие № 3

Использование сценариев для создания «карусели» из блоков (div). На основании сайта-витрины организовать просмотр одиночных записей из раздела «Акции» или «Наши работы\услуги\товары» в виде «карусели».

3. Веб-дизайн

3.1. Основы веб-дизайна

Сетка

С помощью сетки можно обеспечить взаимное выравнивание элементов. Сетка добавляет на страницу невидимый скелет – шаблон, который можно использовать для размещения элементов таким образом, что контент выглядит аккуратным и упорядоченным. Выравнивание элементов по базовой сетке сделает дизайн аккуратным и цельным.

В дизайне можно использовать любое число колонок, но чаще всего применяется сетка на 12 колонок.

Приведем примеры CSS-фреймворков, использующие сетку:

- Bootstrap (https://getbootstrap.com/docs);
- Foundation (https://get.foundation/);
- Skeleton (http://getskeleton.com/);
- mini.css (https://minicss.us/);
- PureCSS (https://purecss.io/).

Так же в самом CSS существует компонент CSS Grid который позволяет легко выравнивать элементы по сетке без использования CSS-фреймворка.

Типографика

Шрифт может быть красивым, но не читабелен, например, сайты с тонкими и бледно-серыми шрифтами, а бывает так, что текст читается легко, но шрифты непривлекательны – это практически все сайты, где используются предустановленные системные шрифты.

Гарнитура – это «семейство», или набор, шрифтов. Например, Arial – это гарнитура, а Arial Bold – это шрифт. В повседневной практике данные термины зачастую используются взаимозаменяемо.

Основное отличие между шрифтами состоит в наличии (serif) или отсутствии (sans serif) у них засечек, представляющих собой мелкие декоративные элементы на концах букв.

Как правило, печатный текст читается легче, если использованы шрифты с засечками, а текст на дисплее более читабелен, если применены рубленые шрифты. Засечки в печатном тексте помогают глазу цепляться за буквы и быстрее двигаться по строчкам, а вот дисплей мелкие засечки воспроизводит плохо, и удобочитаемость снижается. Однако стоит отметить, что с появлением новых экранов с хорошим разрешением это отличие постепенно исчезает.

В типографике пространство между строками текста абзаца называется межстрочным интервалом, а в CSS – высотой строки.

В дизайне следует придерживаться золотой середины – текст сложно читать, если межстрочные интервалы как слишком велики, так и слишком малы. Старайтесь добавлять в текст «воздух», чтобы повысить удобочитаемость и не перегружать страницу, но при этом не затруднять чтение. Значение межстрочного интервала 1,6 является стандартным множителем для установки межстрочного интервала. Например, если размер шрифта 12px, то высота строки 19,2px, или, если размер шрифта 14px, то высота строки 22,4px.

Кернинг – это процесс изменения расстояния между двумя стоящими рядом буквами, а межбуквенный интервал – это заданное расстояние между буквами определенного шрифта. Изменение межбуквенного интервала влияет на расстояние между всеми соседними буквами в тексте, в то время как кернинг изменяет расстояние только

для одной пары букв. CSS позволяет настраивать расстояние между буквами, используя свойство «межбуквенные пробелы» (letterspacing).

Крупные заголовки смотрятся интереснее при уменьшении межбуквенных интервалов, а небольшой текст может выиграть от их увеличения.

Рассмотрим некоторые важные принципы, которые необходимо помнить при работе с типографикой.

Избегайте выравнивания по ширине или по центру. Выровненный по ширине текст – это такой, который полностью вписывается в полосу (колонку). В этом случае текст выравнивается по левому и правому полю одновременно.

Выровненный по ширине, создает ряд не очень красивых визуальных эффектов:

- при растягивании строки до полного формата полосы между словами могут появиться несуразно большие пробелы;

- большие пробелы между словами могут выстраиваться вертикально друг за другом на нескольких строках, образуя визуально неприглядные «реки» пустого пространства.

Текст, выровненный по центру, может подойти для заголовков, но в случае сомнений лучше выравнивать все строки по левому краю относительно сетки. Тем самым вы облегчите чтение.

Абзац будет намного сложнее читать, если его текст содержит больше 75 или меньше 45 символов в строке.

Для выбора шрифтом рекомендуется использовать шрифты Google Fonts (https://fonts.google.com/?subset=cyrillic). Ресурс позволяет скачивать шрифты и добавлять к html-страницам веб-сайта.

Используйте в дизайне не более двух гарнитур. Выбирайте одну гарнитуру для заголовков и вторую для основного текста. Создавать дополнительные стили в тексте можно, например, с помощью жирного шрифта, курсива, верхнего регистра.

Для подбора шрифтов существуют специальные онлайн сервисы:

- Fontjoy (https://fontjoy.com/) – сервис по подбору шрифтов, который имеет режим генератора, и ручной подбор сочетаний шрифтов;

- Font Combinator (https://www.fontcombinator.app/);

- Design AI (https://designs.ai/fonts/).

Цвет

Избегайте выбора цвета для дизайна случайным образом. Ограничьтесь двумя-четырьмя цветами для элементов сайта, так вы сделаете сайт менее перегруженным. При построении цветовой палитры старайтесь не поддаваться желанию применить сразу все яркие цвета. Используйте, к примеру, преимущественно серый или более натуральные оттенки, добавив один яркий акцент, который позволит выделить важные элементы, не создавая хаоса в общем дизайне.

Существует множество онлайн сервисов, позволяющих выбрать готовую палитру. Например, веб-серввис Material Design Palette (https://www.materialpalette.com/) автоматически отображает образец дизайна с выбранными цветами, что очень помогает визуализировать, как два цвета работают вместе.

Белое пространство, иначе называемое негативным (negative space), представляет собой просто пустое пространство (не обязательно белого цвета). Негативное пространство создает визуальную иерархию между различными элементами интерфейса,

что помогает указывать пользователям на стратегические точки взаимодействия. Более того, негативное пространство сокращает информационный беспорядок на странице и делает контент легким на восприятие.

3.2. Инструменты для веб-дизайна

Онлайн редактор Photopea

Бесплатный онлайн графический редактор Photopea позволяет редактировать фотографии, накладывать эффекты, убирать задний фон и выполнять любые действия с графикой.

Графический редактор доступен онлайн:

- https://www.photopea.com/;
- https://photopea.ru/.

Фавикон FAVICON.ICO

Favicon – это сокращение от favorites icon, то есть иконка для избранного. Это может быть логотип сайта или любое другое изображение, которое позволяет сделать ресурс узнаваемым. Первоначально фавикон служил для визуального выделения сайтов в закладках. Сейчас он отображается практически везде, где это возможно: во вкладках и на пустом начальном экране десктопных и мобильных браузеров, в адресной строке и на странице поисковой выдачи рядом с URL веб-сайта.

Чтобы фавикон начал отображаться, на вкладке нужно подключить файл с графическим изображением фавикона. Для подключения нужно добавить в <head> тег <link> со следующими атрибутами.

Листинг 3.1. Пример подключения фавикона.

```
<head>
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
```

Следует обратить внимание на две детали: размер и расположение. Нужно подключать фавикон в формате .ico в размере 16×16. Файл фавикона в формате .ico обязательно нужно размещать в корневой папке проекта. Веб-сервер всегда ищет favicon.ico в корневой папке проекта и пытается ее подключить к сайту.

Современный веб-браузер умеет работать с векторными фавиконами и этот формат более предпочтительный: меньший вес, наилучшее качество, поддерживает смену тем, не нужно указывать размер. Подключение точно такое же, как и для favicon.ico, только нужно добавить тип.

Листинг 3.2. Пример подключения векторного SVG-фавикона.

```
<head>
  <link rel="icon" href="images/favicons/icon.svg" type="image/svg+xml">
</head>
```

Для создания фавикона используйте графические редакторы или специальные сервисы, например, Favicon (https://www.favicon.by/) – сервис для создания favicon формата .ico. Можно загрузить готовую картинку или нарисовать самостоятельно, используя инструменты графического редактора.

Другие сервисы для создания фавикон:

- https://www.favicon.cc/;
- https://favicon.io/.

Лабораторное занятие №4

Формат изображений ICO. На основании сайта-витрины дополнить веб-страницу изображением favicon.ico.

Дизайн в Figma

Figma – это графический онлайн-редактор. В нем можно создать прототип сайта, интерфейс приложения. В веб-сервисе можно работать из браузера или скачать приложение Figma на компьютер.

Официальный сайт https://www.figma.com/downloads/.

Главная страница авторизованного пользователя в Figma представляет собой рабочий стол, где хранятся все проекты, доступные для редактирования. Чтобы создать собственный дизайн, в верхней правой части экрана нажмите кнопку «+ Design file».

В новом файле пользователю доступны три области: панель инструментов, рабочая область, панель слоев и элементов (Layers, Assets) слева, и панель дизайна и прототипа (Design, Prototype) справа.

В верхней части расположен горизонтальный ряд инструментов. Первый инструмент в этом ряду называется Main menu (Основное меню) и содержит следующие команды:

- Back to files – используется для перехода на стартовую страницу авторизованного пользователя со списком файлов (проектов).
- File – содержит команды (действия) по созданию нового файла (New design file), сохранению на жесткий диск локальной версии файла в формате .fig (Save local copy…), вставке файла графического изображения в текущий проект (Place image…).
- Edit – содержит команды по копированию и вставке объектов проекта, свойств объектов, копированию цвета объектов, а так же по поиску и замене объектов.
- View – содержит настройки и способы отображения объектов проекта, вспомогательных и инструментов для совместной работы в команде.
- Object – содержит команды по группировке объектов, расположению объектов на рабочей области.
- Text – содержит настройки по форматированию текстовых объектов.
- Arrange – содержит инструменты по упорядочению и расположению объектов относительно друг друга в рамках рабочей области.
- Preferences – дополнительные возможности по точной привязке объектов друг к другу.

Панель инструментов представлена следующим набором инструментов:

- Move (Перемещение), Scale (Масштабирование);
- Frame (Фрейм), Section, Slice;
- Rectangle, Line и другие геометрические объекты;
- Pen (Перо), Pencil (Карандаш);
- Text (Текст);
- Resources (Ресурсы);
- Hand (Рука);
- Add comment (Добавить комментарий).

Фрейм (Frame) или артборд (англ. artboard) – основной элемент дизайна в Figma. Это законченный документ, который может быть страницей сайта или экраном мобильного приложения. Фрейм необходим для того, чтобы создать дизайн проекта, состоящего из элементов, с которыми будут взаимодействовать пользователи.

При выборе инструмента Frame в правой части на панели Design открывается список Frame с вариантами типов устройств, из которого можно выбрать подходящий вариант для будущего приложения (Phone, Desktop):

- iPhone SE (ширина экрана 320px);
- Android (шина экрана 360px);
- MacBook Air (ширина экрана 1280px);
- Desktop (ширина экрана 1440px).

Файл проекта может содержать несколько фреймов. Поскольку в среднем пользователи проводят большую часть времени с мобильного телефона, важно отдавать приоритет мобильной версии сайта и начинать проектирование дизайн веб-приложения с мобильной версии. В последующем дизайн-макеты масштабируются под планшетные и десктопные устройства.

При проектировании интерфейса следует уделять внимание его удобству в использовании и информативности, чтобы пользователю требовалось минимум времени для того, чтобы получить нужную информацию или сделать заказ.

После создания фрейма можно воспользоваться панелью Design (Дизайн) для настройки параметров фрейма. Поскольку любой веб-сайт представляет собой протяженный контент, который доступен при скроллинге (прокрутке), то для созданного фрейма желательно установить вертикальный размер H (height) фрейма в пять раз больше выбранного размера экрана, например, для мобильной версии вертикальный размер фрейма будет составлять 3200px и для десктопной версии 5120px. Этот размер в последующем можно изменить по мере необходимости, для того, чтобы можно было отобразить весь дизайн-макет будущего веб-сайта.

Модульная сетка (Layout grid) в Figma помогает упорядочить все элементы дизайна во фрейме. Чтобы легко адаптировать дизайн от одного устройства к другому, используйте колоночную модульную сетку (Columns). Для настройки сетки нажмите «+» в блоке Layout grid на панели Design.

Настройку параметров колоночной модульной сетки можно производить по одному из следующих правил:

- Material Design – рекомендует использовать 4 колонки, с промежутками (расстояниями между колонками) Gutter равными 16px и полями Margin равными 16px. Подробнее описано на сайте Material Design https://m2.material.io/design/layout/responsive-layout-grid.html#columns-gutters-and-margins.
- Bootstrap – рекомендует использовать 12 колонок, с промежутками Gutter равными 30px и полями Margin равными 15px. Подробнее о модульной сетке описано на сайте Bootstrap https://getbootstrap.com/docs/4.0/layout/grid/.

Колоночная модульная сетка позволяет упорядочить горизонтальное расположение объектов на экране.

Для оценки вертикального ритма экранов и примерной оценки того, что попадает на первый экран мобильного устройства при загрузке веб-сайта, а так же для оценки количества экранов, которые требуются для отображения всего контента веб-сайта, следует добавить горизонтальную модульную сетку (Rows).

Настройка параметров Height и Gutter горизонтальной модульной сетки для вертикального ритма экранов устанавливается одинаковым и равно высоте экрана

мобильного устройства, например, 640px для Android Small, 568px для iPhone SE, 832px для MacBook Air, 1024px для Desktop.

Для оценки вертикально ритма расположения элементов на странице используется горизонтальная модульная сетка с параметрами настроек аналогичными настройкам соответствующей вертикальной сетке заданной для фрейма.

По умолчанию в Figma доступна библиотека шрифтов Google Fonts (https://fonts.google.com/). Создание текстового объекта сопровождается созданием текстового слоя на панели Layers.

Настройка параметров текстового объекта выполняется на панели Design. Для текста доступны стандартные параметры:

- Fonts (Гарнитура),
- начертание,
- размер,
- Align (Выравнивание) строчек абзаца текстового блока,
- межстрочный интервал (Line height),
- отступы между абзацами (Paragraph spacing),
- межсимвольный интервал (Letter spacing),
- и прочие параметры (Type settings).

Цвет текстового блока определяется параметром Fill.

Для того чтобы добавить одно или несколько изображений в макет воспользуйтесь одним из следующих способов:

- Main menu > File > Place image…;
- инструмент Shape tools > Place image/video;
- просто перетащите изображение с компьютера на рабочую область Figma.

Изображения для сайта лучше всего подготавливать в специальных графических редакторах, например, Adobe Photoshop или подобных ему программах. Так же можно воспользоваться онлайн редактором изображений Photopea (https://www.photopea.com/).

Фрейм (Frame) объединяет объекты внутри себя. Когда создается дизайн внутри фрейма, на панели Layers (Слои) будут добавляться слои. Слои в Figma – это содержимое фрейма: объекты, текст, фотографии.

Можно группировать отдельные объекты в один объект для удобства работы. Для этого необходимо выделить объекты либо по имени на панели Layers (Слои) или на рабочей области фрейма.

Для того чтобы выделить объекты по именам на панели Layers (Слои) нажмите и удерживайте клавишу CTRL на клавиатуре и щелкайте указателем мыши по названиям объектов на панели Layers (Слои).

Для того чтобы выделить объекты на рабочей области фрейма необходимо нажать и удерживать на клавиатуре клавишу SHIFT и щелкать указателем мыши на объекты расположенные во фрейме.

После того как объекты выделены, их можно объединить, выбрав команду Group selection одним из способов:

- Main menu > Object > Group selection;
- вызвать контекстное меню на выделенных объектах;
- нажать комбинацию «горячих клавиш» CTRL + G.

Отдельные объекты или группы объектов можно заблокировать. Блокировка (Lock) – это когда объект остается на своем месте, и вы не можете его перемещать. Блокировка удобна для того, чтобы случайно не сдвинуть размещенный правильно объект.

Для того чтобы блокировать объект или группу объектов необходимо предварительно этот объект выделить и выбрать команду Lock (Блокировать) одним из способов:

- Main menu > Object > Lock/Unlock selection;
- вызвать контекстное меню на выделенных объектах;
- нажать комбинацию «горячих клавиш» CTRL + SHIFT + L.

Для того чтобы разблокировать (Unlock) объекты нужно повторить действия перечисленные выше.

Компоненты в Figma помогают применять изменения к группе элементов. Это экономит время при изменениях макета.

Чтобы превратить объект или группу объектов в компонент, выделите их и нажмите Create Component (CTRL + ALT + K).

Организовать хранение компонентов для небольших проектов, например, сайта из десятка страниц, можно в отдельном фрейме-контейнере. Например, создайте новый фрейм, назовите его «Components» и вложите в него родительские компоненты.

Дополнительные материалы:

- https://tilda.education/articles-figma
- https://tilda.education/courses/web-design/basicsteps/
- https://tilda.education/#design
- https://tilda.school/
- https://www.figma.com/
- https://help.figma.com/hc/en-us/sections/4405269443991-Figma-for-Beginners-tutorial-4-parts-
- https://www.figma.com/resource-library/design-basics/

Идеи для дизайна

Ознакомьтесь с лучшими идеями веб-дизайна, чтобы создать красивый и функциональный сайт. Улучшите пользовательский опыт и привлекайте больше посетителей с помощью этих веб-дизайнерских идей:

- Студия Атемия Лебедева – https://www.artlebedev.ru/web/
- Бюро Горбунова – https://bureau.ru/projects/websites/
- Студия Борового – https://www.db.by/portfolio/landings/
- Сообщество Awwwards – https://www.awwwards.com/
- Сообщество CSSDesignAwards – https://www.cssdesignawards.com/
- Tilda Publishing – https://tilda.cc/ru/tpls/
- TemplateMonster – https://www.templatemonster.com/ru/landing-page-templates-type/
- Как разработать дизайн и код персонального веб-сайта – https://habr.com/ru/articles/480200/

Контрольная точка №2

Веб-дизайн веб-страницы средствами графического редактора – демонстрация веб-дизайна интернет-витрины. Нарисовать в Figma макет мобильной версии лендинг страницы. Нарисовать дизайн, ориентируясь на аналог, пример лендинг веб-страницы для мобильной версии. В качестве примера взять дизайн решение в Интернете (см. Идеи для

дизайна). При выполнении задания используйте цветовую схему, шрифты, компоновку контента на странице. Структура веб-страницы должна содержать как минимум три раздела, например, Контакты, Услуги (Товары, работы и т.п.), Новости (Анонсы, Скидки, Акции и т.п.).

4. Шаблоны проектирования в веб-дизайне

4.1. Шаблоны bootstrap

Фреймворк Bootstrap (Twitter Bootstrap) – свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Основными инструментами Bootstrap являются:

- Сетки – заранее заданные размеры колонок, которые можно сразу использовать. Например, ширина колонки 140px относится к классу .span2 (.col-md-2 в третьей версии фреймворка), который можно использовать в CSS-описании документа.
- Шаблоны – фиксированный или адаптивный шаблон документа.
- Типографика – инструмент описания шрифтов, определение некоторых классов для шрифтов, таких как код, цитаты и т. п.
- Медиа – инструмент, предоставляющий некоторое управление изображениями и видео.
- Таблицы – средства оформления таблиц, вплоть до добавления функциональности сортировки.
- Формы – CSS-классы для оформления форм и некоторых событий, происходящих с ними.
- Навигация – CSS-классы оформления для панелей, вкладок, перехода по страницам, меню и панели инструментов.
- Алерты – инструмент оформления диалоговых окон, подсказок и всплывающих окон.

Для использования фрейморка Bootstrap необходимо подключить к веб-странице таблицу его стилей.

Листинг 4.1. Пример подключения таблицы стилей.

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwI
H" crossorigin="anonymous">
```

Многие из компонентов Bootstrap требуют использования JavaScript для работы. В частности, для них требуются плагины JavaScript разработанные Bootstrap и Popper.js. Для этого необходимо поместить блок <script> обязательно в конце страниц, прямо перед закрывающим тегом </body>, следующего содержания.

Листинг 4.2. Пример подключения JavaScript библиотеки Bootstrap.

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
```

Итоговый вариант html-документа с подключенным фреймворком Bootstrap выглядит следующим образом.

Листинг 4.3. Пример html-шаблона с использованием фреймворка Bootstrap

```
<!doctype html>
<html lang="ru">
  <head>
    <!-- Обязательные метатеги -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwI
H" crossorigin="anonymous">
    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
    <!-- Необязательный JavaScript; выберите один из двух! -->
    <!-- Вариант 1: пакет Bootstrap с Popper -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
    <!-- Вариант 2: отдельные JS для Popper и Bootstrap -->
    <!--
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"
integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js"
integrity="sha384-
0pUGZvbkm6XF6gxjEnlmuGrJXVbNuzT9qBBavbLwCsOGabYfZo0T0to5eqruptLy"
crossorigin="anonymous"></script>
    -->
  </body>
</html>
```

Bootstrap разрабатывается в первую очередь для мобильных устройств. Это стратегия, в которой код сначала оптимизируем для мобильных устройств. Адаптивность верстки веб-страниц осуществляется с помощью медиа-запросов CSS. Чтобы обеспечить надлежащий рендеринг и масштабирование касанием для всех устройств используется мета тег <meta> адаптивного окна просмотра описанный в <head>: <meta name="viewport" content="width=device-width, initial-scale=1">.

Дополнительные материалы:

- официальный сайт Bootstrap – https://getbootstrap.com/docs/5.3/getting-started/introduction/

- Bootstrap по-русски – https://bootstrap5.ru/docs/getting-started/introduction

4.2. Шаблон Carousel

Шаблон Carousel (карусель) – это шаблон проектирования для демонстрации серии слайдов. Каждый слайд демонстрируется обычно несколько секунд, пока не сменится следующим. Смена слайдов может осуществляться как автоматически через равные промежутки времени, так и вручную.

Чаще всего в качестве слайдов используются изображения. Слайд в Bootstrap карусели может быть представлен не только изображением, но и текстовым контентом. Кроме этого при создании слайда можно использовать html-разметку.

Листинг 4.4. Пример html-разметки шаблона Carousel.

```
<!-- Bootstrap 4 -->
<div id="carousel" class="carousel slide" data-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      <!-- html разметка первого слайда -->
    </div>
    <div class="carousel-item">
      <!-- html разметка слайда -->
    </div>
    …
  </div>
</div>
```

Обратите внимание, что класс **active** должен быть добавлен к одному из слайдов. Атрибут data-ride="carousel" запускает автоматическую смену слайдов карусели после загрузки страницы.

Листинг 4.5. Пример html-разметки шаблона Carousel с элементами управления.

```
<!-- Bootstrap 4 -->
<div id="carousel" class="carousel slide" data-ride="carousel">
  <div class="carousel-inner">
    … см. Листинг 4.4
  </div>
  <!-- Элементы управления -->
  <a class="carousel-control-prev" href="#carousel" role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Предыдущий</span>
  </a>
  <a class="carousel-control-next" href="#carousel" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Следующий</span>
  </a>
</div>
```

Поведение кнопок «Предыдущий» и «Следующий» определяется в карусели с помощью атрибута data-slide.

Листинг 4.6. Пример html-разметки шаблона Carousel с индикаторами слайдов.

```
<!-- Bootstrap 4 -->
<div id="carousel" class="carousel slide" data-ride="carousel">
  <!-- Индикаторы -->
  <ol class="carousel-indicators">
    <li data-target="#carousel" data-slide-to="0" class="active"></li>
    <li data-target="#carousel" data-slide-to="1"></li>
    <li data-target="#carousel" data-slide-to="2"></li>
  </ol>
  <div class="carousel-inner">
    … см. Листинг 4.4
  </div>
  <!-- Элементы управления -->
  … см. Листинг 4.5
</div>
```

Карусель с элементами управления и (или) с индикаторами слайдов должна иметь id. Элементы управления и индикаторы слайдов должны иметь атрибут data-target (или href для ссылок), значение которого должно указывать на карусель (т.е. состоять из # и id). Атрибут data-slide-to добавляет к карусели возможность дополнительной навигации по слайдам с помощью индикаторов. Атрибут data-slide-to в качестве значения содержит порядковый номер (индекс) слайда. Отсчет слайдов в карусели ведется с нуля. Если необходимо чтобы при клике на индикатор пользователь перешел, на третий слайд, то к индикатору необходимо добавить атрибут data-slide-to со значением 2.

Листинг 4.7. Пример html-разметки слайда в виде изображения.

```
<div class="carousel-item">
  <img src="..." alt="...">
</div>
```

Компонент Carousel автоматически не нормализует размеры изображений (слайдов) и могут потребоваться дополнительные утилиты или стили, чтобы привести содержимое к необходимому размеру.

Листинг 4.8. Пример html-разметки слайда с текстом (надписями, подписями).

```
<div class="carousel-item">
  <div class="carousel-caption">
    …
  </div>
</div>
```

К слайдам можно добавить надписи. Осуществляется это посредством добавления к каждому слайду, некоторого элемента, например, div, с классом carousel-caption. При необходимости эти надписи можно с помощью классов display отображать на одних экранах и скрывать на других.

Добавление к карусели класса carousel-fade позволяет изменить анимацию перехода на анимацию появления.

Листинг 4.9. Пример html-разметки шаблона Carouse с классом carousel-fade для анимации появления слайда при переходе.

```
<div id="carousel" class="carousel slide carousel-fade" data-ride="carousel">
```

```
  …
</div>
```

Настройка карусели осуществляется с помощью dataатрибутов.

Атрибут data-interval устанавливает количество времени в миллисекундах (паузу) между автоматической сменой слайдов карусели. Значение по умолчанию (миллисекунд): 5000, т.е. 5 секунд. Если данному параметру указать значение false, то карусель не будет выполнять автоматическую смену слайдов.

Атрибут data-keyboard определяет должна ли карусель реагировать на события клавиатуры. Значение по умолчанию: true.

Атрибут data-pause. Значение по умолчанию: "hover". Если параметр pause имеет значение "hover", то смена слайдов, при нахождении курсора над ней, не будет выполняться. А при покидании курсора смена слайдов будет опять возобновляться. Если параметру pause установить значение false, то нахождение курсора над каруселью не будет останавливать автоматическую смену слайдов. На устройствах с сенсорным экраном, при значении "hover", пауза будет устанавливаться на касание. Но после касания, карусель начнет сменять слайды только после времени, равное 2 интервалам (по умолчанию, равное 10 секундам).

Атрибут data-ride имеет значение по умолчанию: false. При значении false, запуск автоматической смены слайдов произойдет только после того, как пользователь вручную (с помощью элементов управления или индикаторов) перейдет к другому слайду. Если в качестве этого параметра установить значение "carousel", то смена слайдов запустится автоматически сразу после загрузки страницы.

Атрибут data-wrap. Значение по умолчанию: true. Данный параметр определяет должна ли смена слайдов зацикливаться. Т.е. после последнего слайда осуществляться переход к первому при движении next (Следующий) и после первого к последнему при движении prev (Предыдущий). Если это не нужно, то значение параметра wrap необходимо установить равное false.

Дополнительные материалы:

- Carousel – https://getbootstrap.com/docs/4.0/components/carousel/
- Карусель – https://getbootstrap.su/docs/5.0/components/carousel/
- Bootstrap - Carousel (карусель) – https://itchief.ru/bootstrap/carousel

4.3. Шаблон Navbar

Шаблон Navbar (Навигационная панель) в Twitter Bootstrap представляет собой навигационную панель (navs), которая реализована в виде вкладок (tabs) и кнопок (pills).

Для создания навигационной панели используется структура маркированного списка (
) к которому добавляется базовый класс .nav.

Затем создаются пункты меню (
), которые будут являться вкладками (tabs) или кнопками (pills).

Листинг 4.11. Пример html-разметки шаблона базового горизонтального меню.

```
<ul class="nav">
  <li class="nav-item">
    …элемент меню
  </li>
  …
</ul>
```

По умолчанию выравнивания пунктов меню выполняется по левому краю. Для выравнивания по правому краю или по центру страницы используются классы .justify-content-end и .justify-content-center соответственно.

Листинг 4.12. Пример использования класса .justify-content-center для выравнивания элементов меню по центру в шаблоне Navbar.

```
<!-- Centered nav -->
<ul class="nav justify-content-center">
```

Листинг 4.13. Пример использования класса .justify-content-end для выравнивания элементов меню по правому краю в шаблоне Navbar.

```
<!-- Right-aligned nav -->
<ul class="nav justify-content-end">
```

В качестве элемента навигации может быть использован любой html-контейнер, например, Гипертекстовая ссылка.

Листинг 4.14. Пример использования шаблона Navbar.

```
<nav class="navbar navbar-expand navbar-dark bg-dark" aria-label="Пример навигационной панели">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Кликни, чтобы развернуть</a>
    <div class="collapse navbar-collapse" id="navbarsExample02">
      <ul class="navbar-nav me-auto">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Активная ссылка</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Ссылка 2</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

Листинг 4.15. Пример использования шаблона Navbar.

```
<div class="collapse navbar-collapse justify-content-md-center"
id="navbarsExample10">
  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link active" aria-current="page" href="#">Активная ссылка</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Ссылка 2</a>
    </li>
    <li class="nav-item">
      <a class="nav-link disabled" aria-disabled="true">Нерабочая ссылка</a>
    </li>
  </ul>
</div>
```

Дополнительные материалы:

- Navbar – https://getbootstrap.com/docs/4.0/components/navbar/
- Панел навигации – https://getbootstrap.su/docs/5.0/components/navbar/

4.4. Шаблон Album

Пример использования шаблона Album можно посмотреть на официальном сайте Bootstrap - https://getbootstrap.com/docs/5.3/examples/album/

4.5. Шаблон Card(-s)

Card – это компонент Bootstrap, который позволяет оформить контент в виде карточки. Карточка имеет гибкую структуру:

- с заголовком (футером) или без него;
- с использованием картинки (ее можно расположить в верхней или нижней части) или без нее;
- с произвольным количеством элементов и их расположением в основной части.

Кроме этого карточке можно придать нужное цветовое оформление: цвет фона, текста и границ.

Листинг 4.16. Пример html-разметки карточки шаблона Card.

```
<div class="card">
  <div class="card-body"><!-- Начало текстового контента -->
     Этот текст расположен внутри card-body...
  </div><!-- Конец текстового контента -->
</div><!-- Конец карточки -->
```

Основные классы, используемые в шаблоне Card:

- .card-body (текстовый контент);
- .card-img-top или card-img-bottom (изображение);
- .card-img-overlay (текстовый контент над изображением);
- .card-header (заголовок);
- .card-footer (футер);
- .list-group (список).

Внутри блоков шаблона Card можно использовать предопределенные классы. Например, с классом .card-body можно использовать следующие классы:

- .card-title (для оформления заголовка);
- .card-subtitle (для стилизации подзаголовка);
- .card-text (для стилизации текста);
- .card-link (для оформления ссылок).

Дополнительные материалы:

- Card – https://getbootstrap.com/docs/5.3/components/card/

4.6. Шаблон Container(-s)

Контейнеры являются самым основным элементом макета в Bootstrap и необходимы при использовании по умолчанию разметкой Grid в Bootstrap. Контейнеры используются для содержания, заполнения и (иногда) центрирования содержимого внутри них. Хотя контейнеры могут быть вложенными, для большинства макетов вложенный контейнер не требуется.

Фреймворк Bootstrap предоставляет следующие классы для форматирования контейнеров:

- .container устанавливает max-width для каждой контрольной точки;

- .container-fluid, который равен width: 100% во всех контрольных точках;

- .container-{breakpoint}, для которого ширина width: 100% до указанной контрольной точки.

Класс .container по умолчанию является адаптивным контейнером фиксированной ширины – это означает, что его максимальная ширина меняется в каждой контрольной точке.

Листинг 4.17. Пример использования контейнера.

```
<div class="container">
  <!-- Контент здесь -->
</div>
```

Дополнительные материалы:

- Containers – https://getbootstrap.su/docs/5.0/layout/containers/
- Контейнеры – https://getbootstrap.su/docs/5.0/layout/containers/

4.7. Шаблон Grid

Для адаптивного дизайна используются следующие граничные размеры экранов (Layout Breakpoints) и infix-добавки в названиях класса:

- Extra small для экранов <576px, без infix;
- Small для экранов ≥576px, infix=sm;
- Medium для экранов ≥768px, infix=md;
- Large для экранов ≥992px, infix=lg;
- Extra large для экранов ≥1200px, infix=xl;
- Extra extra large для экранов ≥1400px, infix=xxl.

Для контейнеров (Layout Containers) используются следующие классы:

- .container – этот класс задает max-width для ширины контейнера в рамках каждого граничного размера экрана (breakpoints);

- .container-{breakpoint} – этот класс задает width: 100% до тех пор, пока граничный размер экрана не превысит указанного значения {breakpoint};

- .container-fluid – этот класс задает width: 100% для всех граничных размеров.

Листинг 4.18. Пример html-разметки с использованием класса .container.

```
<div class="container">
  …
</div>
```

Сетка (Layout Grid) подразумевает использование контейнеров (container), рядов или строк (row), и колонок (column).

Листинг 4.19. Пример html-разметки с использованием grid.

```
<div class="container text-center">
  <div class="row">
    <div class="col">
      Колонка 1
    </div>
    <div class="col">
      Колонка 2
    </div>
    <div class="col">
      Колонка 3
```

```
    </div>
  </div>
</div>
```

Класс .col-sm-6 используется для задания ширины колонки для малых экранов с размерами от 576px и до 768px. В частности, .col-sm-6 означает, что столбец будет занимать половину доступной ширины контейнера, в котором он находится, для небольших экранов.

Листинг 4.20. Пример html-разметки с использованием класса .col-sm-6.

```
<div class="container">
  <div class="row">
    <div class="col-md-6 col-sm-12">
      Этот столбец будет занимать половину доступной ширины для средних экранов и всю доступную ширину для маленьких экранов.
    </div>
    <div class="col-md-6 col-sm-12">
      Этот столбец также будет занимать половину доступной ширины для средних экранов и всю доступную ширину для маленьких экранов.
    </div>
  </div>
</div>
```

Дополнительные материалы:

- Grid system – https://getbootstrap.com/docs/5.0/layout/grid/
- Система сеток – https://getbootstrap.su/docs/5.0/layout/grid/

Лабораторное занятие №5

Шаблон адаптивной верстки. На основании сайта-витрины дополнить шаблон и правила адаптивного отображения содержимого веб-страницы для различных экранов.

Контрольная точка №3

Адаптивная верстка – демонстрация применения шаблонов адаптивного поведения интернет-витрины.

Литература

Ашманов, И. С. Оптимизация и продвижение сайтов в поисковых системах / И. С. Ашманов, А. Иванов. - М. : Питер М, 2013. - 464 с.

Берд, Д. Веб-дизайн. Руководство разработчика / Д. Берд. - М. ; СПб. ; Нижний Новгород : Питер, 2012. - 224 с.

Бердышев, С. Н. Искусство оформления сайта : практическое пособие / С. Н. Бердышев. - 2-е изд. - М. : Дашков и К., 2013. - 148 с.

Васильев, В. В. Практикум по Web-технологиям : для студентов высших учебных заведений, обучающихся по специальности 071201 "Библиотечно-информационная деятельность" / В. В. Васильев, Н. В. Сороколетова, Л. В. Хливненко. - М. : ФОРУМ, 2009. - 416 с.

Гарднер, Л. Д. Разработка веб-сайтов для мобильных устройств / Л. Д. Гарднер. - СПб. : Питер, 2013. - 448 с.

Голомбински, К. Добавь воздуха! Основы визуального дизайна и графики, веб и мультимедиа / К. Голомбински, Р. Хаген. - М. ; СПб. ; Нижний Новгород : Питер, 2013. - 272 с.

Гото, К. Веб-дизайн: книга Келли Гото и Эмили Котлер / К. Гото, Э. Котлер. - 2-е изд. - СПб. : Символ-Плюс, 2007. - 416 с.
Дронов, В. А. HTML 5, CSS 3 Web 2.0. Разработка современных Web-сайтов : руководство / В. А. Дронов. - СПб. : БХВ-Петербург, 2014. - 416 с.
Макнейл, П. Веб-дизайн. Книга идей веб-разработчика / П. Макнейл. - М. ; СПб. ; Нижний Новгород : Питер М, 2014. - 288 с.
Макнейл, П. Настольная книга веб-дизайнера. Все, что вы должны знать о дизайне для интернета / П. Макнейл. - М. ; СПб. ; Нижний Новгород : Питер М, 2013. - 264 с.
Никсон, Р. Создаем динамические веб-сайты с помощью PHP и MySQL, JavaScript, CSS : руководство / Р. Никсон. - 2-е изд. - М. ; СПб. ; Нижний Новгород : Питер, 2013.
Нильсен, Я. Веб-дизайн: книга Якоба Нильсена : научное издание / Я. Нильсен. - СПб. : Символ-Плюс, 2001. - 512 с.
Осборн, Т. Веб-дизайн для недизайнеров / Т. Осборн. - М. ; СПб. ; Минск : Питер, 2022. - 176 с.
Робсон, Э. Изучаем HTML, XHTML и CSS / Э. Робсон, Э. Фримен. - 2-е изд. - СПб. : Питер, 2020. - 720 с.
Фримен, Э. Изучаем программирование на JavaScript / Э. Фримен, Э. Робсон. - СПб. : Питер, 2020. - 640 с.