Лабораторная работа №1

Вёрстка веб-страниц с использованием HTML/CSS на основе готового шаблона

Задание:

1. Выбрать готовый шаблон для оформления контентных страниц сайта.

2. Придумать тематическое содержание (контент) страниц сайта.

3. Разработать на основе выбранного шаблона 3-4 (в зависимости от количества человек в подгруппе) html-страниц, используя **CSS**-стили и **HTML5**- модель верстки.

4. На каждой странице в блоке уникального содержимого (теги <main>) реализовать различные стилизованные элементы: список, форма ввода, изображение с текстом, таблица с текстовым наполнением, предусмотреть блок ссылок на соцсети в «подвале».

5. Реализовать переходы на созданные страницы через навигационное меню (простыми тегами для ссылок).

Теоретическая часть.

1. Ознакомиться с методами верстки: прямая, табличная, блочная, БЭМ (блок-элемент-модификатор).
2. Специфика HTML 5, основные теги для разметки страницы.
3. Основы CSS: применение, подключение, селекторы и отношения.

Основы HTML5 верстки:

https://developer.mozilla.org/ru/docs/Learn/HTML/Introduction_to_HTML/Document_and_website_structure

https://html5book.ru/vyorstka-stranicy-sayta

Основы CSS: https://html5book.ru/css-css3

Инструментарий для редактирования HTML и CSS:

1. NotePad++
2. Atom
3. VSCode + плагин HTML CSS Support

Инструменты для работы с шаблонами стилей:

1. Paint.Net: https://paintnet.ru/download/
2. Gimp: https://www.gimp.org/downloads/
3. Photoshop

Методические рекомендации.

1. Подготовка рабочей директории. Перед началом работы создаем файловую структуру сайта – индексная страница с отдельными папками для изображений и css-стилей (рис. 1):

Рис. 1 – Файловая структура в рабочей папке

2. Проработка контента. После создания файловой структуры определяемся с тематикой контентных страниц и выбираем стиль из готового шаблона в формате **PSD**. Шаблоны можно взять, например, здесь:

http://www.pcklab.com/templates

Файл PSD – это растровый графический проект программы Photoshop, сохраненный без сжатия и содержащий дополнительную информацию, такую как: иерархия слоев, прозрачность элементов и т.д.

Одной из популярных свободно распространяемых программ для работы с данным форматом является Gimp. Скачайте данное ПО и откройте PSD-файл. Структура слоев проекта будет представлена в правом нижнем меню (рис. 2):

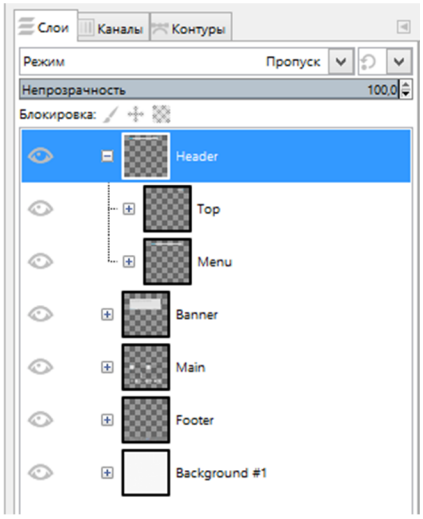

Рис. 2 – Дерево слоёв в Gimp

Используя данную структуру, «нарезаем» необходимые для реализации html-страниц графические элементы. Основные из них:

- Логотипы.

- Фоновые изображения для заголовочного блока, основной части и подвала (Footer). Если фоновые изображения имею повторяющийся рисунок (сетка, линии, повторяющийся узор и т.п.), то необходимо «нарезать» фон таким образом, чтобы его можно было корректно совместить при горизонтальном или вертикальном копировании.

- Маркеры, иконки меню и соцсетей.

Для сохранения графических элементов необходимо выделить соответствующий слой в меню и нажать сочетание клавиш Ctrl+C, далее необходимо открыть программу Paint.Net и скопировать содержимое через Ctrl+V или Shift+Insert. В Paint.Net нажимаем в панели инструментов «Обрезать по выделению» и сохраняем данные в формате PNG через меню «Файл»→ «Сохранить как…» (рис. 3).

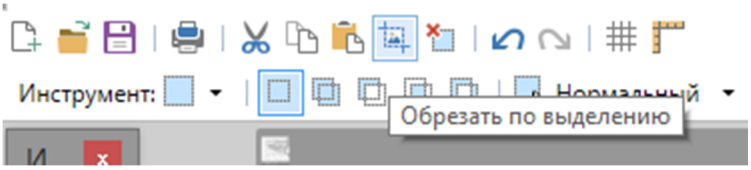

Рис. 3 – Обрезание полотна по размеру скопированного изображения

Сохраняем все графические элементы в папке **images** (если графических элементов много, группируйте их в отдельных папках, например, изображения с

логотипами в папке images/logo, общие для всех страниц графические элементы в папке images/common и т.п.).

3. Реализация HTML-кода. Базовый код для html-страницы:

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Это заголовок страницы</title>
        <link rel="stylesheet" href="css/styles.css">
    </head>
    <body>
        Здесь будет реализация уникального контента страницы
    </body>
</html>
```

Строка <link rel="stylesheet" href="css/styles.css"> в заголовке подключает css-файл с описанием стилей веб-страницы.

Базовый «каркас» **HTML5 документа** (рис. 4):

```
<body>
    <header></header>
    <nav></nav>
    <main>
        <aside></aside>
        <article></article>
    </main>
    <footer></footer>
</body>
```

- **<header>** представляет собой группу вводного содержимого. Если <header> является дочерним элементом <body>, то он определяет общий заголовок для содержимого веб-страницы, т.е. здесь может быть логотип и / или название сайта или компании. Если же <header> расположен как дочерний элемент <article> или <section>, то он определяет конкретный заголовок для отдельного раздела.

- **<nav>** содержит основные функции навигации для страницы. Вторичные ссылки и т. д. не входят в навигацию.

- **<main>** предназначен для содержимого, *уникального для этой страницы*. Используйте <main> только *один* раз на странице и размещайте прямо внутри <body>. В идеале он не должен быть вложен в другие элементы.

- **<article>** содержит блок связанного содержимого (уникальный контент), который имеет смысл сам по себе без остальной части страницы (например, один пост в блоге, статья и т.д.). Каждый элемент <article> должен быть выделен с помощью добавления заголовка (элементы <h1>-<h6>) в качестве дочернего элемента:

```
<article>
    <h2>Прогноз погоды</h2>
    <p>Температура: +5 °C</p>
    <p>Давление: 760 мм рт. ст.</p>
    <p>Влажность: 2%</p>
</article>
```

- **<section>** подобен <article>, но больше подходит для группирования одной части страницы, которая представляет собой одну часть функциональности (например, мини-карту или набор заголовков статей и сводок). Также считается хорошей практикой начинать каждый раздел с заголовка. В зависимости от контекста вы можете разбить <article> на несколько <section> или, наоборот, <section> на несколько <article>.

- **<aside>** содержит контент, который не имеет прямого отношения к основному содержимому, но может содержать дополнительную информацию, косвенно связанную с ним: словарь, биография автора, связанные ссылки и т. д.).

- **<footer>** представляет собой группу конечного контента для страницы, т.н. «подвал». Обычно также содержит логотип компании и контактные данные (ссылки на соцсети, почтовый адрес и т.п.)

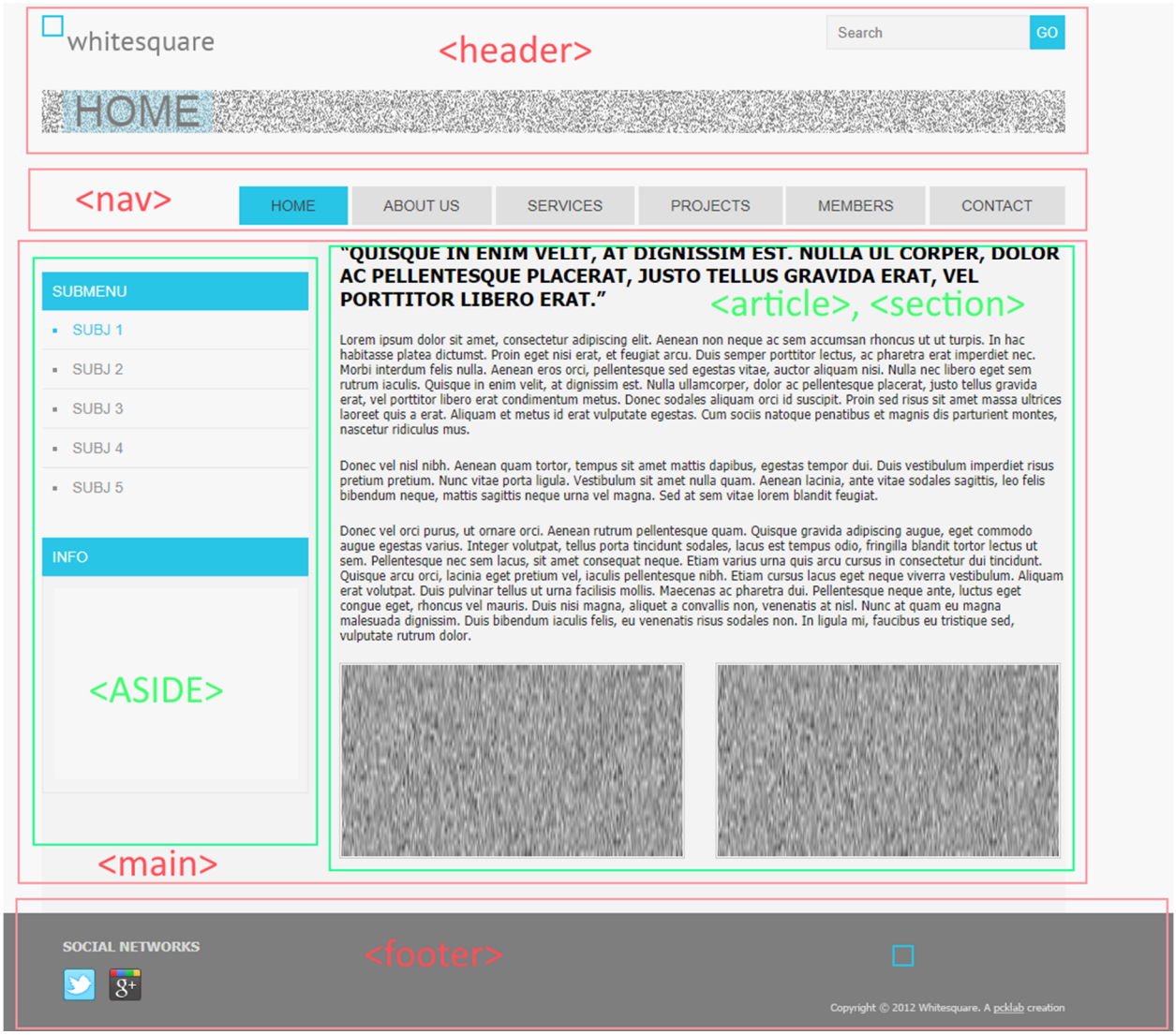

Рис. 4 – Разметка базовых блоков на странице

Главная условие хорошо сверстанной страницы – возможность перемещения блоков в любое место. Например, у вас есть блок новости. Теперь в блок новости нужно добавить блок погоды. В данном случае, если у вас используется каскадная верстка с большими уровнями вложенности (.class div div и т.д.) и сложными правилами приоритетов (например, inline-стиль или директива !important), то, скорее всего, придется переписывать верстку для погоды и новостей. Это может стать проблемой в большом проекте, где изменение стиля одного из блоков может «сломать» стили других блоков или нарушить структуру web-страницы.

Если во время работы веб-страницы требуется изменить стиль какого-то элемента «на лету» или выводить внутри него какой-либо текст (т.е. менять состояние этого элемента, например, с помощью js-кода), то можно подключать стиль по id элемента, чтобы в дальнейшем обращаться к элементу через метод getElementById,

параметром которого служит имя идентификатора. В остальных случаях для стилей лучше использовать class-селекторы.

При использовании в CSS следует учитывать, что идентификаторы обладают высоким приоритетом по сравнению с классами.

Если приоритет у классов одинаковый, тогда, в случае противоречия будут задействованы те свойства, которые указаны в стиле ниже.

В названиях классов желательно закладывать семантический смысл, а не особенность реализации. Например, если таким образом мы выделяли более важные части текста, то стоило назвать класс important:

```
.important {
  color: red;
  font-style: italic;
}
```

Теперь при изменении цветов или любых других деталей этого стиля не нужно придумывать новое название, т.к. смысл не поменяется, поменяется лишь конкретная реализация.

Также при создании имен CSS-стилей можно придерживаться нейминга в рамках БЭМ-подхода.

БЭМ (Блок, Элемент, Модификатор) — компонентный подход к веб-разработке. В его основе лежит принцип разделения интерфейса на независимые блоки. Он позволяет легко и быстро разрабатывать интерфейсы любой сложности и повторно использовать существующий код, избегая «Copy-Paste».

В рамках данного подхода используется следующий шаблон для названия стилей:

имя-блока__имя-элемента_имя-модификатора_значение-модификатора

Подробнее с данным подходом можно ознакомиться здесь:

https://ru.bem.info/methodology/naming-convention/

Результатом выполнения задания являются стилизованные HTML-страницы и отчет, содержащий следующую информацию:

1) Конспект необходимого теоретического материала для выполнения лабораторной работы.
2) Скриншоты HTML-страницы с обозначением разметки базовых тегов (аналогично рис. 4).
3) Таблица, содержащая названия базовых блоков (по скриншотам) и соответствующие им имена CSS-селекторов.

Для успешной защиты лабораторной работы студенты должны предоставить файлы HTML-страниц, CSS-стилей и отчет.

Требования к оформлению отчета:

Способ выполнения текста должен быть единым для всей работы. **Шрифт** –

Times New Roman, кегль 14, **межстрочный интервал** – 1,5, **размеры полей**: левое – 30 мм; правое – 10 мм, верхнее – 20 мм; нижнее – 20 мм. Сокращения слов в тексте допускаются только общепринятые.

Абзацный отступ (1,25) должен быть одинаковым во всей работе. **Нумерация страниц** основного текста должна быть сквозной. Номер страницы на титульном листе не указывается. Сам номер располагается внизу по центру страницы или справа.