Федеральное государственное бюджетное образовательное учреждение высшего образования
УЛЬЯНОВСКИЙ ГОСУДАРСВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ МАТЕМАТИКИ, ИНФОРМАЦИОННЫХ И АВИАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра телекоммуникационных технологий и сетей

А.А. Булаев

Методические указания по выполнению лабораторных работ по дисциплинам «Web-программирование» и «Мультимедиа-технологии»

Учебное пособие

Ульяновск
2019

УДК 004.42, 004.43

Рецензент: к.т.н., доцент кафедры информатики УлГПУ им И.Н. Ульянова Лукьянов Владимир Анатольевич

Методические указания по выполнению лабораторных работ по дисциплинам «Web-программирование» и «Мультимедиа-технологии» учебное пособие / А.А. Булаев. – Ульяновск: УлГУ, 2019. - 45 с.

Данное учебное пособие ориентировано на курсы «Web-программирование», «Мультимедиа-технологии», «Программирование в Интернет», «Компьютерная геометрия и графика» и «Программирование на языке Python». В пособие включены необходимые для изучения основ web-программирования теоретические материалы, варианты лабораторных работ и описан процесс их выполнения.

Пособие предназначено для студентов факультета математики, информационных и авиационных технологий.

УДК 004.42, 004.43

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время web-технологии являются одной из самых популярных технологий разработки приложений. Web-программирование в современном понимании позволяет разрабатывать не только многостраничные web-сайты со статичной информацией, но и целые системы с возможностью реализации Интернет-магазинов, социальных сетей и Интернета вещей. Благодаря web-технологиям многие пользователи «всемирной паутины» имеют возможность получить необходимую информацию, товары и услуги в режиме, близком к реальному времени.

Знание таких технологий как PHP, JavaScript, Python, HTML5 и CSS позволяет без труда реализовывать как браузерные приложения, так и приложения для современных мобильных устройств и планшетов. Существует большое количество оболочек, внедряющих HTML-код в мобильные и настольные приложения.

Таким образом, созданный web-ресурс, взаимодействующий с базой данных и реализующий широкие функциональные возможности, достаточно просто адаптировать для различных операционных систем и устройств.

Веб-программирование осуществляется при помощи специальных программных средств - скриптов. Эти программные средства подразделяются на два основных вида: серверные и клиентские. Серверные скрипты выполняются на стороне сервера, то есть того компьютера, на котором размещен сайт. Они выполняются еще до загрузки страниц сайта на компьютер пользователя. В свою очередь, клиентские скрипты выполняются на компьютере клиента уже после загрузки страницы с сервера и не требуют ее дополнительной перезагрузки.

Языки программирования, на которых выполняются и те, и другие скрипты различны. Некоторые из языков используются только для создания серверных скриптов, другие - только для клиентских, а многие языки - для тех и других.

В настоящем пособии представлен теоретический материал для выполнения лабораторных работ по web-технологиям студентами кафедры «Телекоммуникационных технологий и сетей».

Раздел 1. Основы World Wide Web (WWW)

Сетевая модель OSI

Для единого представления данных в сетях между различными устройствами международная организация по стандартам ISO разработала базовую модель связи открытых систем OSI (Open System Interconnection). Эта модель описывает правила и интерфейсы передачи данных по сети. Основными элементами модели являются уровни, прикладные процессы и физические средства соединения. Каждый уровень модели OSI выполняет определенную задачу в процессе передачи данных по сети.

Каждому уровню модели OSI соответствует протокол, обеспечивающий взаимодействие устройств друг с другом. Примерами протоколов прикладного уровня являются: HTTP, FTP; транспортного: TCP, UDP; сетевого: IPv4, IPv6; канального: Ethernet, PPP. На физическом уровне рассматривается среда передачи данных: витая пара, оптоволоконный кабель и др.

Протокол HTTP

HTTP (HyperText Transfer Protocol) - протокол прикладного уровня передачи гипертекста, основанный на клиент-серверной архитектуре.

Структура протокола HTTP.

HTTP-сообщение состоит из трёх блоков:

- строка запроса, в которой указан метод передачи, URL-адрес ресурса, версия протокола HTTP;
- заголовки, в которых указываются параметры передачи сообщения;
- тело сообщения – сами данные.

Формат строки запроса клиента:

```
Метод URI Протокол/Версия_протокола
```

Пример строки запроса клиента:

```
GET /index.html HTTP/4.1
```

Формат строки ответа сервера:

```
Протокол/Версия_протокола Код_состояния [Пояснение]
```

Пример ответа сервера на описанный выше запрос клиента:

```
HTTP/4.1 200 Ok
```

Методы протокола HTTP

Метод HTTP — последовательность символов, указывающая на основную операцию над ресурсом. Наиболее используемые методы:

- **GET –** метод запроса содержимого web-ресурса;

```
GET /путь_к_файлу?параметр1=значение1&параметр2=значение2 Протокол/Версия_протокола;
```

- **POST –** метод передачи пользовательских данных web-ресурсу (могут отправляться данные из форм регистрации и авторизации пользователей, изображения, другие медиа-файлы);
- **OPTIONS –** метод определения возможностей и параметров соединения web-сервера;

```
OPTIONS * HTTP/4.1;
```

- другие методы**: HEAD, PUT, PATCH, DELETE, TRACE, CONNECT**.

Коды состояний отображают результат выполнения запроса сервером и представляют собой трёхзначное число, в котором первая цифра указывает на класс состояния, а другие две – на порядковый номер.

Поддерживаемые современными серверами и клиентами коды состояний:

- **1xx** – информационные коды, сообщающие о процессе передачи без передачи ответа на сервер (**100** – Continue (продолжить), **101** – Switching Protocols (переключение протоколов), **102** – Processing (идёт обработка));
- **2xx** – коды успеха обработки запроса (**200** – OK (успешно), **201** – Created (создано), **202** – Accepted (принято), **204** – No Content (нет содержимого), 206 – Partial Content (частичное содержимое));
- **3xx** – коды перенаправлений (**300** – Multiple Choices (множественный выбор), **301** – Moved Permanently (перемещено навсегда), **304** – Not Modified (не изменялось));
- **4xx** – коды ошибок клиента (**401** – Unauthorized (Неавторизован), **402** – Payment Required (Требуется оплата), **403** – Forbidden (Запрещено), **404** – Not Found (Не найдено), **405** – Method Not Allowed (Метод не поддерживается), **406** – Not Acceptable (Не приемлемо), **407** – Proxy Authentication Required (Требуется аутентификация прокси));
- **5xx** – коды ошибок сервера (**500** – Internal Server Error (Внутренняя ошибка сервера), **502** – Bad Gateway (Плохой шлюз), **503** – Service Unavailable (Сервис недоступен), **504** – Gateway Timeout (Шлюз не отвечает))

Заголовок HTTP – это строка, содержащая разделённую двоеточием пару вида «параметр-значение».

```
Server: Apache/2.2.11 (Win32) PHP/5.5.0
Last-Modified: Tue, 10 Jan 2018 12:12:12 GMT
Content-Type: text/plain; charset=utf-8
Content-Language: ru
```

Структура HTML-документа

Язык HTML - это стандартизированный язык разметки, который описывает правила оформления и набор структурных и семантических элементов разметки (тегов).

Разработкой спецификаций языка HTML занимается *Консорциум всемирной паутины* (W3C).

Версии спецификаций языка HTML:

- HTML 1.0 (официально не существует, у каждого браузера была своя реализация);
- HTML 2.0 – стандарт 22 сентября 1995 года;
- HTML 3 – 1996 год;
- HTML 3.2 – 14 января 1997 года;
- HTML 4.0 – 18 декабря 1997 года;
- HTML 4.01 – 24 декабря 1999 года;
- HTML 5 – разработан и принят W3C совместно с сообществом WHATWG;
- HTML 5.1 – 17 декабря 2012 года;
- HTML 5.2 – 14 декабря 2017 года;
- XHTML 1.0 – семейство языков разметки на основе XML, расширяющих возможности HTML 4;
- XHTML 1.1.

Особенности спецификаций XHTML:

- имена тегов и атрибутов записываются строчными буквами;
- все теги должны быть закрыты. Если тег не имеет закрывающего элемента, то в конце ставится символ / (например, <img />);
- логические атрибуты записываются в виде атрибут="атрибут" (checked="checked");
- кодировкой по умолчанию является UTF-8.

Любой HTML-документ содержит такие основные конструкции как: теги <html>, <head>, <body> и специальный элемент <!doctype>.

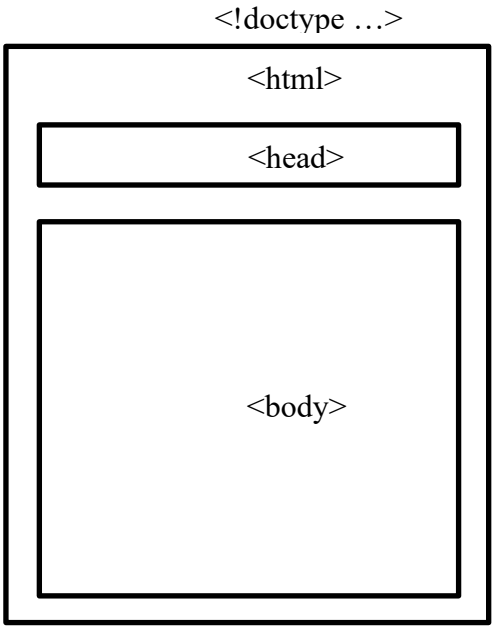


Рисунок 1. Структура HTML-документа

Элемент **<!DOCTYPE>** предназначен для указания типа текущего документа (версии HTML/XHTML).

Мета-тег HTML – это элемент разметки HTML, описывающий свойства документа. Назначение мета-тега определяется набором его атрибутов, которые задаются в теге <meta>.

Тег (html-тег, тег разметки) – управляющая символьная последовательность, которая задает способ отображения гипертекстовой информации.

Пример HTML-страницы:

```
<!DOCTYPE HTML>
<html>
 <head>
   <title>Тег META</title>
   <meta charset="utf-8">
 </head>

 <body>
   <div>Текст</div>
 </body>
</html>
```

Наиболее используемые теги:

- **<html></html> –** контейнер гипертекста;
- **<head></head>** – контейнер заголовка документа;
- **<body></body>** – контейнер тела web-страницы;
- **<title></title>** – название web-страницы;
- ** ** – структурный блок;
- ** ** – текстовый абзац;
- **<a>** – гиперссылка (пример: *Текст*);
- **
** – маркированный список;

- **
** – нумерованный список;
- **
** – элемент списка;
- **<table></table>** – таблица;
- **<tr></tr>** – строка таблицы;
- **<td></td>** – ячейка таблицы;
- **<img>** – изображение (пример: *<img src="путь_к_файлу" alt="описание">*);
- **<form></form>** – форма (используется для отправки данных на сервер);
- **
** – перенос строки.

Атрибуты – это пара вида “свойство = значение”, уточняющие соответствующий тег:

```
<тег атрибут="значение">...</тег>
```

Таблицы стилей

CSS – язык описания внешнего вида документа, оформленного языком разметки HTML.

Стиль – это совокупность правил, применяемых к элементу гипертекста и определяющих способ его отображения. Стиль включает все типы элементов дизайна: шрифт, фон, текст, цвета ссылок, поля и расположение объектов на странице.

Таблица стилей – это совокупность стилей, применимых к гипертекстовому документу.

Каскадирование – это порядок применения различных стилей к веб-странице. Браузер, поддерживающий таблицы стилей, будет последовательно применять их в соответствии с приоритетом.

Другой аспект каскадирования – **наследование** – означает, что если не указано иное, то конкретный стиль будет применен ко всем дочерним элементам гипертекстового документа. Например, если вы примените определенный цвет текста в теге , то все теги внутри этого блока будут отображаться этим же цветом.

Формат записи CSS:

```
селектор {
 свойство1: значение1;
 свойство2: значение2;
}
```

Селектор – это элемент, к которому будут применяться назначаемые стили. В качестве селектора может использоваться тег, класс (с использованием точки в начале: ".имя_класса"), идентификатор (с использованием знака «решётка» в начале: "#имя_идентификатора"), эти элементы могут применяться как вместе, так и по отдельности. Для задания свойств сразу нескольким селекторам достаточно их записать через запятую.

Формат записи селектора:

```
тег#идентификатор.класс
```

Свойство определяет одну или несколько характеристик селектора (шрифт, размер, отступ и т.д.).

Значение – это фактические числовые или строковые константы, определяющие свойство селектора.

Пример записи стиля для тега **<p>**:

```
p {
 text-align: left;
 text-indent: 20px;
 font-family: Serif;
 font-size: 12px;
}
```

Для добавления таблицы стилей на HTML-страницу необходимо использовать тег **<style>:**

```
<style type="text/css">
h1 {
      color: #333366;
   }
</style>
```

Для добавления таблицы стилей в виде отдельного файла используется тег **<link>:**

```
<link rel=stylesheet href="имя.css" type="text/css">
```

Раздел 2. Разработка страниц на языке PHP

PHP (PHP Hypertext Preprocessor) – это широко используемый язык сценариев общего назначения с открытым исходным кодом.

Значительным отличием PHP от какого-либо кода, выполняющегося на стороне клиента, например, JavaScript, является то, что PHP-скрипты выполняются на стороне сервера.

Программа на *PHP* – это набор команд (инструкций), который встраивается в HTML-код посредством конструкции <?php ?>.

Переменная в PHP обозначается знаком доллара ($), за которым следует ее имя. Имя переменной чувствительно к регистру, т.е. переменные $my_var и $My_var различны. Например:

```
<?php
  $my_var = "значение переменной";
  $My_var = "значение переменной 2";
?>
```

В PHP имеется возможность задания переменной по ссылке. Для этого значение должно иметь имя, т.е. оно должно быть представлено какой-либо другой переменной. Чтобы указать, что значение одной переменной присваивается другой переменной по ссылке, нужно перед именем первой переменной поставить знак амперсанд (&). В последующем примере значения переменных $first и $second будут всегда совпадать:

```
<?php
  $second = &$first;
?>
```

Для хранения постоянных величин, т.е. таких величин, значение которых не меняется в ходе выполнения скрипта, используются ***константы***. Такими величинами могут быть математические *константы*, пароли, пути к файлам и т.п. Основное отличие *константы* от *переменной* состоит в том, что ей нельзя присвоить значение больше одного раза и ее значение нельзя аннулировать после ее объявления. Кроме того, у *константы* нет приставки в виде знака доллара и ее нельзя определить простым присваиванием значения.

```
<?php
define("Имя_константы",
         "Значение_константы",
        [Нечувствительность_к_регистру])
?>
```

Кроме констант, объявляемых пользователем, в PHP существует ряд констант, определяемых самим интерпретатором. Например, константа *__FILE__* хранит имя файла программы (и путь к нему), которая выполняется в данный момент, *__FUNCTION__* содержит имя функции, *__CLASS__* – имя класса, *PHP_VERSION* – версия интерпретатора PHP.

PHP поддерживает **восемь** простых *типов данных*. В *PHP* не принято явное объявление *типов переменных*. Предпочтительнее, чтобы это делал сам *интерпретатор* во *время выполнения* программы в зависимости от контекста, в котором используется *переменная*.

Четыре скалярных типа:

- *boolean (логический)*;
- *integer (целый)*;

- *float (с плавающей точкой)*;
- *string (строковый)*.

Два смешанных типа:

- *array (массив)*;
- *object (объект)*.

И два специальных *типа*:

- *resource (ресурс)*;
- *NULL*.

Массив в PHP представляет собой упорядоченную карту – тип, который преобразует значения в ключи. Этот тип оптимизирован в нескольких направлениях и может использоваться как собственно массив, список (вектор), хеш-таблица (являющаяся реализацией карты), стек, очередь и т.д. Языковая конструкция *array* () принимает в качестве параметров пары «*ключ => значение*», разделенные запятыми. Символ => устанавливает соответствие между *значением* и его *ключом*. *Ключ* может быть как *целым числом*, так и *строкой*, а *значение* может быть любого имеющегося в PHP *типа*. Числовой *ключ массива* часто называют индексом. Индексирование *массива* в PHP начинается с нуля. *Значение* элемента *массива* можно получить, указав после имени *массива* в *квадратных скобках ключ* искомого элемента. Если *ключ массива* представляет собой стандартную запись *целого числа*, то он рассматривается как число, в противном случае – как *строка*. Поэтому запись $a["1"] равносильна записи $a[1], так же как и $a["-1"] равносильно $a[-1].

```
1:  <?php
2:  $arr1 = array(5 => 43, 6 => 32,
3:                  7 => 56, "b" => 12);
4:  ?>
```

Объект *– тип данных*, пришедший из объектно-ориентированного программирования (*ООП*). Согласно принципам *ООП*, класс – это набор *объектов*, обладающих определенными свойствами и методами работы с ним, а *объект* соответственно – экземпляр класса. Например, программисты – это класс людей, которые пишут программы, изучают компьютерную литературу и имеют имя и фамилию. Теперь, если взять одного конкретного программиста, Василия Петрова, то он является *объектом* класса программистов, обладает теми же свойствами, что и другие программисты: имеет имя, пишет программы и т.п.

В PHP для доступа к методам *объекта* используется *оператор ->*. Для инициализации *объекта* используется выражение ***new***, создающее в *переменной* экземпляр *объекта*.

Пример создания класса и объекта:

```php
<?php
class Person {
    function know_php() {
            echo "Теперь я знаю PHP";
    }
}
$bob = new Person; // создаем объект
$bob->know_php(); // обучаем его PHP
?>
```

Ресурс – это специальная *переменная*, содержащая ссылку на внешний *источник* (например, соединение с базой данных). *Ресурсы* создаются и используются специальными функциями (например, *mysql_connect*(), *pdf_new*() и т.п.).

Специальное значение ***NULL*** говорит о том, что *переменная* не имеет значения.

Переменная считается *NULL*, если:

- ей была присвоена *константа NULL* ($var = NULL);
- ей еще не было присвоено какое-либо значение;
- она была удалена с помощью *unset*().

Оператор if

Оператор if позволяет выполнять фрагменты кода в зависимости от условия. Структуру оператора *if* можно представить следующим образом:

```php
<?php
if (условие1) {
  выражение1;
} elseif (условие2) {
  выражение2;
} else { // в противном случае
  выражение3;
}
?>
```

В процессе обработки скрипта выражение преобразуется к *логическому типу*. Если в результате преобразования значение выражения истинно (True), то выполняется *блок_выполнения*. В противном случае *блок_выполнения* игнорируется.

Если блок_выполнения содержит несколько команд, то он должен быть заключен в фигурные скобки {}.

Правила преобразования выражения к *логическому типу*:

1. В FALSE преобразуются следующие значения:
    - логическое False;
    - целый ноль (0);
    - действительный ноль (0.0);
    - пустая строка и строка "0";
    - массив без элементов;
    - объект без переменных (подробно об объектах будет рассказано в одной из следующих лекций);
    - специальный тип NULL.
2. Все остальные значения преобразуются в TRUE.

Циклы

Цикл while

Цикл *while* выполняет код, пока соблюдается некоторое условие:

```
<?php
  $d = 2;
  while($d <= 5) {
    echo $d . "<br />";
    $d++;
  }
?>
```

Пока условие выполняется, оператор echo выводит строку, а значение переменной изменяется в сторону увеличения с каждым витком цикла.

При использовании цикла *while* важно помнить об изменении значения условия во избежание создания бесконечного цикла.

Цикл do…while

Отличительной чертой цикла *do...while* является то, что сначала срабатывает код и только после этого проверяется условие. В примере переменной присвоено значение 1, дальше к $d прибавлено 2, то есть значение переменной $d после первого витка цикла будет равно 3; цикл повторяется и заканчивается выводом строки со значением переменной равной 9, так как оператор *while* находится в конце программного кода.

```
<?php
  $d = 1;
  do {
    $d = $d + 2;
    echo $d . "<br />";
  }
  while ($d <= 7);
?>
```

Цикл For

Цикл *for* является циклом со счетчиком и используется для выполнения тела цикла определенное число раз.

```
<?php
  for ($x=0; $x<10; $x++) {
    echo $x;
  }
?>
```

Цикл Foreach

Цикл *foreach* является циклом перебора элементов массива. Команды циклически выполняются для каждого элемента массива, при этом очередная пара «*ключ=>значение*» оказывается в переменных *$ключ* и *$значение.*

При создании цикла есть возможность не задавать переменную для ключа.

```php
<?php
  $names["Лето"] = "Июль";
  $names["Зима"] = "Январь";
  $names["Весна"] = "Апрель";
  $names["Осень"] = "Октябрь";

  foreach ($names as $key => $value) {
    echo "key = $key | value = $value <br>";
  }

  foreach ($names as $value) {
    echo "value = $value <br>";
  }

?>
```

Для любого цикла можно указать конструкции *break* и *continue*, которые позволяют соответственно прервать цикл или пропустить текущую итерацию.

Конструкция Switch

Конструкция *switch* подобно *if else* используется для выбора фрагмента кода, который будет выполнен в случае соблюдения условия. Оператор *case* - определяет условие, *break* - определяет границу участка кода, *default* - выводит некоторое значение, если условия не соблюдаются.

```php
<?php
  $t=date("H");
  switch ($t) {
    case ($t < 06):
      echo "Ночь";
      break;
    case ($t < 10):
      echo "Утро";
      break;
    case ($t < 18):
      echo "День";
      break;
    default:
      echo "Вечер";
  }
?>
```

Раздел 3. Динамический HTML

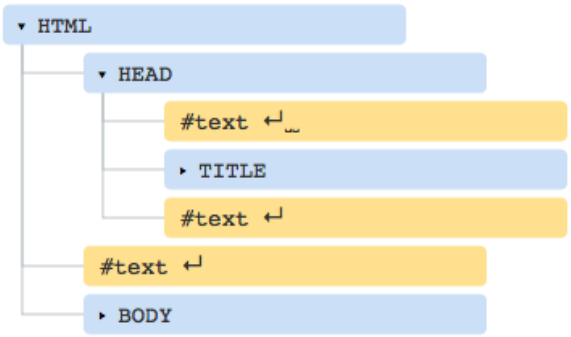
Объектная модель документа (DOM)

Любая страница, написанная с помощью языка разметки HTML, может быть представлена в виде иерархии или дерева тегов.

DOM – это программный интерфейс для HTML и XML документов, который предоставляет структурированное представление документа и определяет то, как эта структура может быть доступна из программ, которые могут изменять содержимое, стиль и структуру документа (например, при помощи языка JavaScript).

В дереве DOM выделяются четыре типа узлов:

- узлы-элементы (node), которые содержат информацию о соответствующих им тегах и включают вложенные в них дочерние теги;
- текстовые узлы (text), которые содержат текстовую информацию и не могут содержать дочерних элементов;
- корневой элемент (document) для входа в дерево DOM;
- комментарий (comment) для отображения комментариев в HTML-коде страницы.



При обработке некорректного HTML-кода браузер автоматически исправляет его для показа и при построении DOM. В частности, всегда будет верхний узел - тег <html>. То же самое касается и тега <body>. Например, если файл состоит из одного слова "Привет", то браузер автоматически «обернёт» его в <html> и <body>.

При генерации DOM браузер самостоятельно обрабатывает ошибки в документе, закрывает теги и др.

Основное назначение DOM - чтение информации из HTML, создание и редактирование элементов.

Узел **HTML** можно получить как document.documentElement, а **BODY** – как document.body. Выбрав элемент, появляется возможность редактирования его атрибутов:

```
1: document.body.style.backgroundColor = 'red';
```

Основные особенности DOM:

- DOM-модель – это внутреннее представление HTML-страницы в виде дерева;
- все элементы страницы, включая теги, текст, комментарии, являются узлами DOM;
- у элементов DOM есть свойства и методы, которые позволяют изменять их.

Язык программирования JavaScript

JavaScript — это динамический язык программирования, который применяется к HTML-документу и обеспечивает динамическую интерактивность на веб-сайтах.

Возможности JavaScript:

- создание, редактирование и удаление элементов HTML-страницы;
- взаимодействие с внешними API (Google, Yandex, Twitter, Facebook, VK, Instagram и т.д.);
- подключение сторонних фреймворков и библиотек для добавления новых возможностей;
- работа с аудио- и видеопотоками, web-камерой, микрофоном;
- создание 2D- и 3D-графики;
- поддержка всеми распространёнными современными браузерами;
- полная интеграция с HTML и CSS.

В JavaScript невозможно:

- взаимодействие с файловой системой;
- запуск программ и прямой доступ к файловой системе;
- взаимодействие с другими вкладками или окнами браузера.

Код на языке JavaScript можно интегрировать в сам HTML-код:

```
<script type="text/javascript">
// код на языке JavaScript;
</script>
```

либо отдельным файлом:

```
<script type="text/javascript" src="main.js"></script>
```

В случае, если тег <script> добавляется в блок <head>, то весь JavaScript-код из этого тега загружается до отображения страницы и уже начинает функционировать при загрузке содержимого блока <body>.

Если тег <script> находится в блоке <body>, то его код начинает функционировать с момента прочтения браузером этого тега. Если до места добавления скрипта вызывается какая-либо функция из него, то браузер сообщит о том, что эта функция не существует.

Основы JavaScript.

Обозначение переменной:

```
var myVariable;
```

Имя переменной может состоять из букв, цифр и знаков $ и _ (подчёркивание). Имя переменной не должно начинаться с цифры. В имени различаются строчные (маленькие) и прописные (большие) буквы. Например, *Alfa, alfa* и *aLFa* – разные переменные. В качестве имён переменных нельзя использовать ключевые слова языка JavaScript. Переменные могут иметь следующие типы.

Типы данных:

- строка (String), пример:

```
var myVariable = "значение";
```

- число (Number), пример:

```
var myVariable = 10;
```

– булевый (Boolean), значения: **true**/**false**, пример:

```
var myVariable = true;
```

– массив (Array), пример:

```
var myVariable = [1, 'Bob', 'Steve', 10];
```

– объект (Object), пример:

```
var myVariable = document.body;
```

В JavaScript тип переменной явно не объявляется. Создают переменную чаще всего присваивая ей начальное значение.

В JavaScript используются три логических оператора: *И, ИЛИ и НЕ*, имеющие обозначения &&, || и ! соответственно. Результат логической операции может иметь одно из двух значений: *true* (истина) и *false* (ложь). Логической величине *true* соответствует число 1, а *false* – 0.

Массивы в JavaScript одновременно рассматриваются и как объекты и как массивы в традиционном понимании. Нумерация элементов массива начинается с нуля. Для обращения к элементу массива используются квадратные скобки. Элементы одного массива могут иметь разный тип.

Комментарии:

```
/*
Пример комментария
*/
// Пример другого комментария
```

Условия:

```
if (iceCream == 'chocolate') {
  alert('Шоколадное мороженое!');
} else {
  alert('Другое мороженое');
}
```

Функции:

```
// Пример функции умножения двух чисел
function multiply(num1, num2) {
  var result = num1 * num2;
  return result;
}

// Запись той же самой функции немного в другом виде
var multiply2 = function(num1, num2) {
  var result = num1 * num2;
  return result;
}
```

События:

```
// Событие на нажатие кнопки <button>
document.querySelector('button').onclick = function() {
  alert('Нажата клавиша!');
}
```

В JavaScript имеется глобальный объект *window*, который содержит в себе характеристики окна браузера, DOM-объект страницы, а также глобальные методы и события, отвечающие за взаимодействие с окном, изменение содержимого, преобразование типов данных и др. Если создать свою функцию и записать её в глобальный объект *window* (например, *window.my_function*), то в дальнейшем её можно вызвать из любого скрипта на данной странице.

События

Изменения в динамических страницах могут происходить либо циклически, либо в ответ на действия пользователя. Во всех случаях сигналом к изменениям служит какое-либо событие. В динамическом HTML события включены в число атрибутов элементов. Следующие события используются большинством элементов:

- *onclick* возникает, когда указательное устройство (мышь) щёлкает на элементе;
- *ondblclick* возникает, когда указательное устройство (мышь) дважды щёлкает на элементе;
- *onmousedown* возникает, когда кнопка указательного устройства (мыши) "нажала" на элемент;
- *onmouseup* возникает, когда кнопка указательного устройства (мыши) отпущена над элементом;
- *onmouseover* возникает, когда указательное устройство (мышь) проходит над элементом;
- *onmousemove* возникает, когда указательное устройство (мышь) перемещается в тот момент, когда находится над элементом;
- *onmouseout* возникает, когда указательное устройство (мышь) убирается с элемента;
- *onkeypress* возникает, когда клавиша нажата и отпущена над элементом;
- *onkeydown* возникает, когда клавиша нажата над элементом.
- *onkeyup* возникает, когда клавиша отпущена над элементом.

Технология AJAX

AJAX – подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером.

В классической модели веб-приложения:

- Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент.
- Браузер формирует и отправляет запрос серверу.
- В ответ сервер генерирует совершенно новую веб-страницу и отправляет её браузеру и т. д., после чего браузер полностью перезагружает всю страницу.

При использовании AJAX:

- Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент.
- Скрипт (на языке JavaScript) определяет, какая информация необходима для обновления страницы.
- Браузер отправляет соответствующий запрос на сервер.
- Сервер возвращает только ту часть документа, на которую пришёл запрос.

- Скрипт вносит изменения с учётом полученной информации (без полной перезагрузки страницы).

Для реализации AJAX можно использовать как «чистый» JavaScript, так и дополнительные фреймворки (библиотеки), такие как: jQuery, MooTools, Prototype, Backbone.js, AngularJS и др.

В качестве примера рассмотрим работу AJAX в библиотеке jQuery.

Метод **load()** – загрузка HTML-кода страницы в указанный элемент текущей страницы.

```
jQuery("div#main").load('example.html');
```

Метод **ajax()** – выполнение асинхронного HTTP-запроса. Данный метод является основным методом для реализации AJAX-запросов и принимает в качестве параметра объект, включающий все настройки запроса.

```
jQuery.ajax({
    url: 'example.html', // URL-адрес запроса
    data: {'username': 'user', 'password': '123456'}, // входные данные
    dataType: "json", // тип загружаемых данных (text, html, json, xml, script)
    success: function (data, textStatus) { // обработчик события об удачной передачи
    },
    error: function (object, textStatus, errorThrown) { // обработчик события об ошибке
    },
    complete: function (object, textStatus) { // обработчик окончания запроса
    }
});
```

Метод **get()** – загрузка данных с использованием GET-запроса.

```
jQuery.get( url [, data] [, success(data, textStatus, jqXHR)] [, dataType] );
```

Метод **post()** – загрузка данных с использованием POST-запроса.

```
jQuery.post( url [, data] [, success(data, textStatus, jqXHR)] [, dataType] );
```

Метод **getJSON()** – загрузка данных в формате JSON.

```
jQuery.getJSON( url [, data] [, success(data, textStatus, jqXHR)] );
```

У всех команд атрибуты, указанные в квадратных скобках, являются необязательными и могут не использоваться при написании исходного кода.

Раздел 4. Графика в HTML

Основы HTML5 Canvas

Canvas - API для рисования, добавленный в HTML и поддерживающийся большинством браузеров. Canvas позволяет рисовать элементы прямо в браузере без использования плагинов (Flash или Java).

Браузер предоставляет прямоугольную область на экране, в котором рисуются линии, фигуры, изображения, текст и другие объекты.

Canvas является частью спецификации HTML5.

Координатная система в Canvas начинается с левого верхнего угла и имеет координаты точки (0,0).

Пример рисования красного прямоугольника с помощью Canvas:

```
<html>
<body>
  <canvas width="800" height="600" id="canvas"></canvas>
  <script>
    var canvas = document.getElementById('canvas');
    var c = canvas.getContext('2d');
    c.fillStyle = "red";
    c.fillRect(100,100,400,300);
  </script>
</body>
</html>
```

В canvas поддерживаются следующие геометрические элементы:

- Прямоугольники
- Линии
- Окружности, дуги
- Различные кривые, эллипс

Следует отметить, что хоть HTML5 Canvas и использует для рисования геометрические примитивы в векторном своём представлении, но результатом изображения является растр, в котором нельзя изменить один конкретный объект, а необходимо полностью перерисовать полотно.

Прямоугольники

В canvas существует несколько способов рисования прямоугольника:

```
strokeRect(x, y, width, height); // пустой прямоугольник с обводкой
fillRect(x, y, width, height)   // Закрашенный прямоугольник
clearRect(x, y, width, height)  // Очистка области на холсте в виде прямоугольника
заданного размера
```

Ниже приведен пример использования 2-х способов:

```
ctx.fillRect(0, 0, canvas.width, canvas.height);
ctx.clearRect(50, 50, 300, 200); // вырезается прямоугольник
```

В результате нарисован большой черный прямоугольник и вырезана область из точки (50, 50) размером 300 пикселей по ширине и 200 пикселей по высоте. Ниже представлено, как это будет выглядеть.

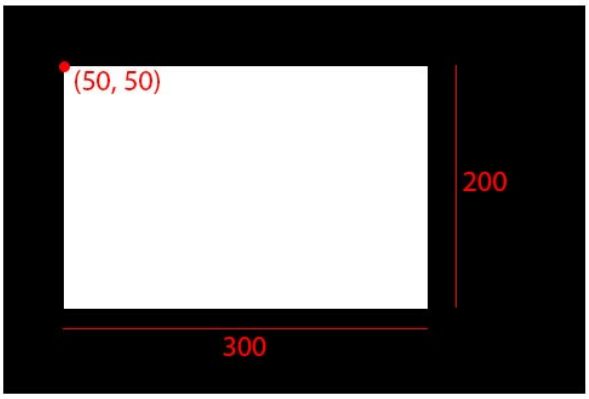


Рисунок 2. Пример рисования прямоугольника на canvas

Ниже представлен пример рисования нескольких прямоугольников, вложенных друг в друга.

```
ctx.strokeRect(5, 5, 138, 138);
ctx.fillRect(10, 10, 128, 128);
for (i = 0; i <= 1; i += 2) {
    for (j = 0; j <= 1; j += 2) {
        ctx.clearRect(10 + i * 64, 10 + j * 64, 64, 64);
        ctx.clearRect(10 + (i + 1) * 64, 10 + (j + 1) * 64, 64, 64);
    }
}
```

Ниже пример, как это выглядит:

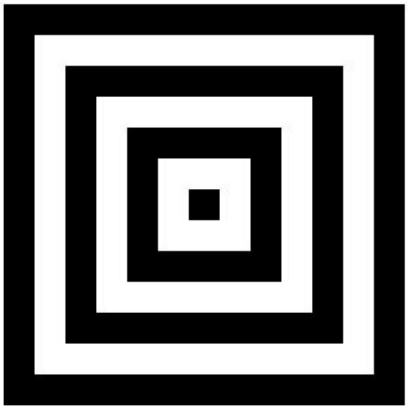


Рисунок 3. Пример рисования вложенных прямоугольников на canvas

Линии, окружности, дуги

Рисование фигур из линий происходит немного сложней. Здесь используется 4 мето-да:

```
/* дословно: начать путь. Используется, чтобы
   начать рисовать фигуры */
beginPath()
/* дословно: закрыть путь. Используется, чтобы
   завершить рисование. Необязательный метод */
closePath()
/* метод обводит фигуру линиями */
stroke()
/* Заливает фигуру сплошным цветом */
fill()
```

Методы для рисования линий:

```
moveTo(x, y) // перемещает "курсор" в указанное место
lineTo(x, y) // ведёт линию из текущей позиции в новое
arc(x, y, radius, startAngle, endAngle, anticlockwise) /* рисование окружности
    startAngle, endAngle - начальный и конечный угол
    anticlockwise - направление */
```

Ниже приведен пример использования данных методов:

```
ctx.beginPath(); // начинаем рисовать
ctx.arc(160, 160, 30, 0, 360, false);
ctx.fill(); // заливаем окружность

ctx.moveTo(160, 0);
ctx.lineTo(200, 120);
ctx.lineTo(320, 160);
ctx.lineTo(200, 200);
ctx.lineTo(160, 320);
ctx.lineTo(120, 200);
ctx.lineTo(0, 160);
ctx.lineTo(120, 120);
ctx.lineTo(160, 0);
ctx.stroke(); // обводим фигуры
ctx.closePath(); // закончили рисовать
```

Результатом рисования является звезда:

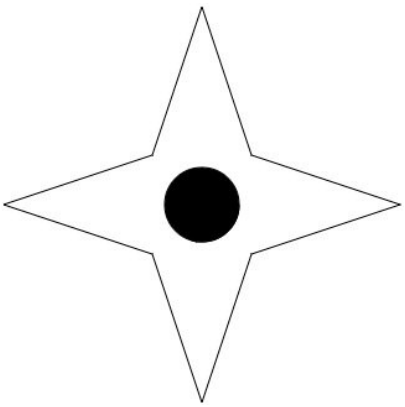


Рисунок 4. Пример рисования в Canvas с помощью линий

Кривая Безье

В canvas возможно рисование фигур с помощью кривых Безье. Для этого используется 2 метода *quadraticCurveTo*, *bezierCurveTo*. Для кривых Безье должна быть задана начальная точка, от которой будет отрисовываться фигура.

Пример кода:

```
ctx.lineWidth = 4;  // ширина линии
ctx.beginPath();
ctx.moveTo(50, 150);
ctx.bezierCurveTo(0, 40, 160, 80, 240, 40); // линия по 3-м точкам
ctx.stroke();
```

Рисунок 5. Пример рисования фигуры кривыми Безье

Цвет линий и заливки в canvas

Для задания цвета есть 2 свойства: *fillStyle* и *strokeStyle*. Задать цвет можно несколькими форматами (названием цвета, HEX-кодом, в формате RGB и в формате RGBA с альфа-каналом):

```
ctx.strokeStyle = "red";
ctx.strokeStyle = "#FF0000";
ctx.strokeStyle = "rgb(255,0,0)";
ctx.strokeStyle = "rgba(255,0,0,1)"
```

Пример задания цвета для разных объектов:

```
ctx.strokeStyle = 'red'; // меняем цвет рамки
ctx.strokeRect(15, 15, 100, 100);
ctx.fillStyle = 'yellow'; // меняем цвет прямоугольника
ctx.fillRect(20, 20, 90, 90);

ctx.strokeStyle = '#00FF00'; // меняем цвет рамки
ctx.strokeRect(125, 15, 100, 100);
ctx.fillStyle = '#5500FF'; // меняем цвет прямоугольника
ctx.fillRect(130, 20, 90, 90);

ctx.strokeStyle = 'rgb(0,0,255)'; // меняем цвет рамки
ctx.strokeRect(235, 15, 100, 100);
ctx.fillStyle = 'rgb(255,0,0)'; // меняем цвет прямоугольника
ctx.fillRect(240, 20, 90, 90);
```

Рисунок 6. Рисование прямоугольников с разными цветами

SVG-графика

Scalable Vector Graphics (масштабируемая векторная графика) - формат изображений на основе текста (XML). Каждое SVG-изображение определено с использованием разметки кода, похожей на HTML. SVG-код может быть включен напрямую в HTML-документ. Все современные веб-браузеры поддерживают SVG. SVG основан на XML, поэтому элементы, не имеющие закрывающего тега, должны быть самозакрывающимися.

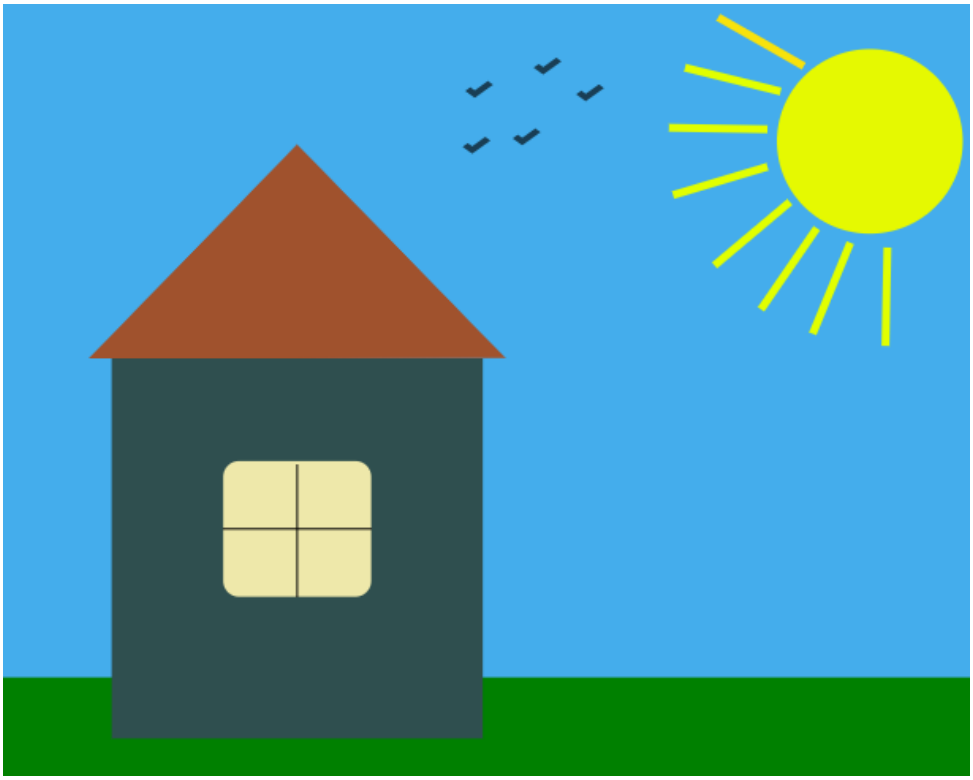


Рисунок 7. Пример изображения с помощью SVG-графики

SVG-изображение является примером векторной графики и поддерживает такие свойства как масштабируемость без потерь качества и привязку событий на JavaScript к любому элементу изображения.

Для создания SVG-изображения необходимо создать сам «холст», у которого необходимо указать ширину и высоту с помощью атрибутов width и height, соответственно. В противном случае SVG-изображение займёт всю площадь блока.

```
<svg width="500" height="50">
</svg>
```

Фигуры

Основными примитивами в SVG являются следующие фигуры: rect, circle, ellipse, line, text и path.

Координатная система в SVG начинается с левого верхнего угла и имеет координаты точки (0,0). Увеличение x происходит слева направо, увеличение *y* - сверху вниз.

Тег ***rect*** рисует квадрат. Квадрат задается четырьмя значениями: x, y - указывают точку верхнего левого угла; width, height - указывают ширину и высоту квадрата.

```
<rect x="0" y="0" width="500" height="50"/>
```

Тег ***circle*** рисует круг. Круг задается тремя величинами: *cx, cy* указывают точку, расположенную в центре описываемой окружности, *r* задает радиус круга. Круг в примере ниже расположен в центре SVG, потому что атрибут *cx* равен 250.

```
<circle cx="250" cy="25" r="25"/>
```

Тег ***ellipse*** задается схоже с *circle*, но предполагается, что радиус задается по двум осям: по *x* и по *y*. Вместо *x* используется атрибут *rx*, вместо *y* - *ry*.

```
<ellipse cx="250" cy="25" rx="100" ry="25"/>
```

Тег ***line*** рисует линию. Для задания координат начала линии используются атрибуты *x1* и *y1*, и *x2* и *y2* - для задания координат конца. Атрибут *stroke* задаёт цвет линии (по умолчанию она невидимая).

```
<line x1="0" y1="0" x2="500" y2="50" stroke="black"/>
```

Стилизация SVG-элементов

Любой элемент SVG имеет чёрную заливку и не имеет рамку. Для изменения этих и других свойства объекта необходимо использовать соответствующие атрибуты.

Общие SVG-свойства перечислены ниже:

- *fill* - заливка. Цветовое значение. Также как и в CSS цвет может быть указан несколькими способами:
  - по имени: orange;
  - значение в шестнадцатиричной системе счисления: #3388aa, #38a;
  - значение в формате RGB: rgb(10, 150, 20);
  - значение в формате RGBA: rgba(10, 150, 20, 0.5).
- *stroke* - рамка. Цветовое значение;
- *stroke-width* - ширина рамки;
- *opacity* - прозрачность. Числовое значение в промежутке от 0.0 (полностью прозрачно) до 1.0 (полностью видимо).
- *и др*.

Для работы с текстовыми элементами дополнительно доступны следующие свойства:

- *font-family* - название семейства шрифтов или общее имя шрифта для выбранного элемента;
- *font-size* – размер шрифта текста;
- *rotate* – угол поворота текста;
- *textLength* – ограничение длины текста;
- *text-anchor* – выравнивание текстовой строки относительно заданной точки (выравнивание слева, по центру или справа);

- *lengthAdjust* – формат изменения текста при заданном ограничении *textLength* (растягивание или сжиматие)
- *и др.*

Раздел 5. Создание Web-приложений на языке Python

При разработке веб-приложений язык Python используется для обработки серверной части сайта аналогично PHP и Ruby. Сами же веб-страницы отображаются с использованием HTML и CSS, а динамическая составляющая на стороне браузера выполняется на JavaScript.

В некоторых фреймворках для разработки web-приложений на языке Python имеется система шаблонов для написания специальных HTML-файлов, которые могут вставлять код Python и взаимодействовать с данными с сервера.

Такой тип взаимодействия называется *full-stack фреймворком*. С его помощью имеется возможность работы с системами, обрабатывающими HTTP-запросы, хранилищами баз данных, шаблонами веб-страниц, запросами маршрутизации и т. д. С другой стороны, есть фреймворки, которые обрабатывают только базовую логику, а для сторонних работ они должны быть объединены с внешними базами данных, шаблонизаторами и т. д.

Web-фреймворки языка Python

Django — один из самых известных и популярных фреймворков с открытым исходным кодом для веб-разработки с использованием Python. Он поставляется с десятками встроенных модулей, взаимодействующих друг с другом. Главный плюс Django – приложение на его основе хорошо масштабируется. Django - фреймворк с открытым исходным кодом, со всеми вытекающими.

Flask — микрофреймворк, который обеспечит базовый уровень возможностей, в то время как основную функциональность на себя должны взять сторонние интегрированные компоненты.

Pyramid – компромисный по функциональности фреймворк, простой и удобный в разработке и достаточный для организации большинства веб-приложений. В этом фреймворке имеется большая библиотека официальных и неофициальных плагинов, реализующих большой спектр задач.

Web.py - это свободно распространяемый web-фреймворк для Python, который реализует большой перечень функций для создания web-приложений.

Рассмотрим пример запуска web-сервера с использованием фреймворка *web.py.*

Наиболее важная часть любого вебсайта - это структура его URL.

Чтобы начать работу с web.py приложением, необходимо создать текстовый файл (например, *code.py*) и ввести для импортирования «web.py»-модуля:

```
1: import web
```

Далее задаётся структура URL адресов:

```
1: urls = (
2:     '/', 'index'
3: )
```

Первая часть - это регулярное выражение, которое соответствует адресам, как */*, */help/faq*, */item/(\d+)*, и др. (*\d+* соответствует последовательности цифр). Скобки вокруг *\d+* используются для получения результата, найденного по этому выражению.

Вторая часть - это имя класса, который будет обрабатывать запрос. Например, *index*, *welcomes.hello* (то есть класс *hello* из модуля *welcomes*). Все остальные элементы выделенного регулярного выражения передаются в класс-обработчик.

То есть, код выше обозначает, что мы хотим обрабатывать адрес / (главная страница) классом с именем *index*. Далее необходимо создать приложение с указанием url адресов.

```
app = web.application(urls, globals())
```

Этой строкой создаётся приложение с адресами, перечисленными выше и использующее классы из глобального пространства имён данного файла.

Пример создания обработчика GET-запроса и POST-запроса:

```
class index:
    def GET(self):
        return "Hello, world! GET"
    def POST(self):
        return "Hello, world! POST"

```

Теперь эти методы будут вызываться каждый раз, когда будет происходить запрос по адресу «/».

Для того, чтобы web-сервер запустился, необходимо добавить следующую строку:

```
if __name__ == "__main__":
    app.run()
```

Для запуска самого web-сервера необходимо в командой строке (консоли) ввести команду:

```
$ python code.py
http://0.0.0.0:8080/
```

Для реализации взаимодействия с базами данных предлагается использовать библиотеку **SQLAlchemy**.

SQLAlchemy — это библиотека на языке Python для работы с реляционными СУБД с применением технологии ORM. Служит для синхронизации объектов Python и записей реляционной базы данных. SQLAlchemy позволяет описывать структуры баз данных и способы взаимодействия с ними на языке Python без использования SQL.

ORM (англ. object-relational mapping, рус. объектно-реляционное отображение) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

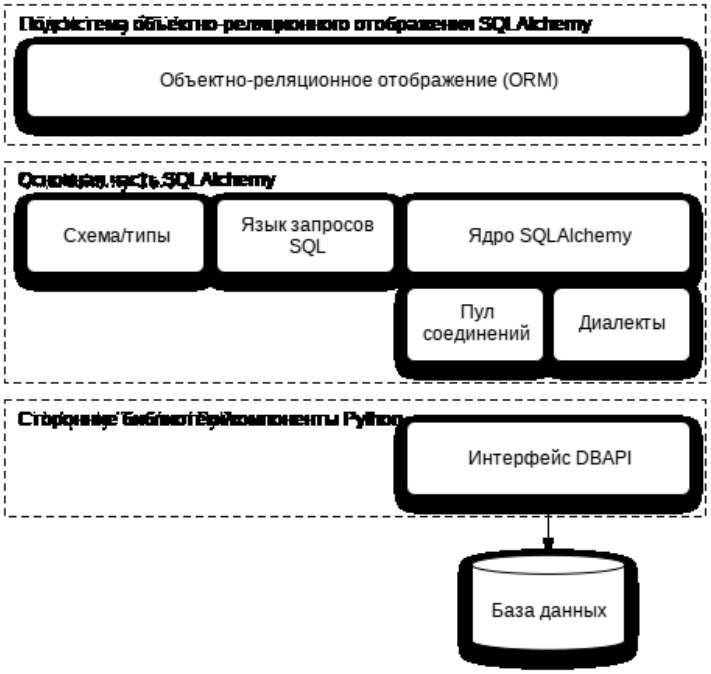


Рисунок 8. Диаграмма уровней SQLAlchemy

Пример подключения и работы SQLAlchemy и СУБД PostgreSQL

```
1:  from sqlalchemy import create_engine
2:
3:  db_string = "postgres://postgres:password@postgres.com:5432/database"
4:
5:  db = create_engine(db_string)
6:
7:  # Создание
8:  db.execute("CREATE TABLE IF NOT EXISTS films (title text, director text, year text)")
9:  db.execute("INSERT INTO films (title, director, year) VALUES ('Star wars', 'George Walton
    Lucas', '1977')")
10:
11: # Чтение
12: result_set = db.execute("SELECT * FROM films")
13: for r in result_set:
14:     print(r)
15:
16: # Обновление
17: db.execute("UPDATE films SET title='Star Wars. Episode IV: A New Hope' WHERE year='1977'")
18:
19: # Удаление
20: db.execute("DELETE FROM films WHERE year='1977'")
```

Полученные данные в дальнейшем можно использовать в запущенном ранее web-сервере.

Вопросы для самоконтроля

1. Семиуровневая модель OSI. Стек протоколов TCP/IP.
2. Протокол HTTP. Структура запросов и ответов. Коды состояния протокола HTTP.
3. Web-серверы. Примеры, особенности использования.
4. Основные понятия языка разметки HTML. Версии HTML. Структура Web-страницы.
5. HTML. Форматирование текста, изменение шрифта, вставка рисунков.
6. Общие атрибуты элементов HTML. Теги заголовка документа. Теги тела документа.
7. Блочные и строчные элементы разметки HTML. Заголовки и абзацы. Нумерованные и маркированные списки.
8. Создание таблиц в HTML. Основные атрибуты таблиц, строк, ячеек.
9. HTML. Способы передачи данных на сервер. Гиперссылки, формы.
10. Определение, назначение, версии каскадных таблиц стилей (CSS).
11. Синтаксис CSS. Идентификаторы, классы, теги.
12. Верстка страниц при помощи CSS. Управление положением на странице (свойства left, top, z-index, position, visibility, overflow).
13. Модель DOM. Уровни. Структура документа.
14. JavaScript, назначение, размещение, основные операторы.
15. Типы данных и классы языка JavaScript. Массивы, хэш-таблицы.
16. Обработка событий при помощи JavaScript. Объект event.
17. Объекты window, document, history, location, screen, navigator.
18. JavaScript. Навигация по дереву документов. Создание узлов. Редактирование дерева элементов.
19. Формат JSON, формат XML.
20. Библиотека jQuery. Обращение к элементам. Создание элементов DOM с помощью jQuery.
21. Базы данных. Системы управления базами данных. Примеры запросов.
22. PHP. Особенности языка. Операторы INCLUDE и REQUIRE. Типы данных, массивы, ассоциативные массивы, классы в PHP.
23. PHP. Обработка данных формы. Методы GET, POST, REQUEST.
24. PHP. Работа с текстовыми файлами. Обработка входных данных. Доступ к базам данных.
25. PHP. Регулярные выражения.
26. Сессии и cokkies в PHP.
27. Технология Ajax. Реализация Ajax с помощью jQuery.
28. Web-сокеты.
29. Способы локального хранения данных.
30. Системы управления содержимым (CMS). Назначение, функции. Примеры.
31. Типы данных, операции, операторы языка Python
32. Операторы цикла, условий, ветвления в Python
33. Особенности ввода/вывода языка Python
34. Списки, кортежи, множества, словари в Python
35. Файловый ввод-вывод на языке Python. Работа с двоичными файлами
36. Работа с базами данных на языке Python
37. Разработка Web-приложений в Python
38. Модули и пакеты в Python

Лабораторные работы

Лабораторная работа №1

Создание HTML-страницы.

ЗАДАНИЕ.

1. Вывести стихотворение А.С. Пушкина построчно с отступами между строфами (не менее 3).
2. Задать каждой строфе отдельный цвет текста.
3. Подчеркнуть любую строку.
4. Сделать курсивом любую строку.
5. Вывести изображение под стихотворением.

МАТЕРИАЛЫ ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ:

1. Для выполнения лабораторных работ будет использоваться OpenServer.
2. Open Server Panel — это локальный веб-сервер для Windows и программная среда, созданная специально для веб-разработчиков.
3. В каталоге «C:\OSPanel\domains\localhost» располагаются файлы для отображения Web-страниц. Основной файл, который загружается браузером, имеет название «index.html». Для просмотра сайта в браузере необходимо открыть: http://localhost/
4. В этом каталоге мы будем размещать все файлы и подкаталоги, созданные во время выполнения лабораторных работ.
5. Создайте или отредактируйте через блокнот или Notepad++ файл *index.html*.

```
<html>
  <head>
    <title>"Программирование в Интернет"</title>
    <meta charset=”utf-8”>
  </head>
  <body>
    <p>Здесь представлены лабораторные работы, выполненные на занятиях по дисциплине "Программирование в Интернет" <Фамилия Имя Отчество>.
    <dl>
      <dt><a href="index.html”>Работа 1</a></dt>
      <dd>Описание.</dd>
      <dt><a href=" путь_к_файлу ">Работа 2</a></dt>
      <dd>Описание.</dd>
    </dl>
  </body>
</html>
```

6. При выполнении работы необходимо указать свою группу и свою фамилию, имя и отчество.
7. Список ссылок оформляем в виде списка определений. Ссылки на другие работы будем оформлять аналогичным образом. В приведённом списке уже указана ссылка на первую

работу.

8. Выведем информацию об интерпретаторе PHP, используемом в программе OpenServer.
9. Создайте файл «work1.php» добавьте в него следующий текст:

```
<html>
  <head>
    <title>"Web-программирование" (Фамилия И.О.) - Работа 1 </title>
    <meta name='viewport' content='width=device-width, initial-scale=1.0' char- set='utf-8'>
  </head>
  <body>
    <?php
      echo "<p><a href='index.php'>К содержанию</a>"; phpinfo();
    ?>
  </body>
</html>
```

10. Выполните все необходимые действия, для того чтобы текст отображался нормально. Сохраните изменения.
11. Выполните задание к данной лабораторной работе

ОТЧЁТ.

1. Для сдачи лабораторной работы необходимо оформить отчёт в виде файла в формате DOC/DOCX.
2. Указать задание к лабораторной работе.
3. Вывести исходный код.
4. Сделать скриншоты, приложить их в отчёт и дать описание.

Лабораторная работа №2

Взаимодействие с файловой системой.

ЗАДАНИЕ.

1. Создать текстовый файл *data.txt*.
2. Записать в текстовый файл данные согласно варианту из таблицы с помощью разделителя (;). В файле должно быть не менее 25 различных записей.

Вариант	**Данные**
1.	Страна, столица, год основания столицы, официальный язык
2.	Фильм, жанр, режиссёр, год выхода, страна, рейтинг
3.	Компьютерная игра, жанр, год выхода, разработчик, издатель
4.	Книга, жанр, автор, год выхода, страна
5.	Операционная система, версия, год выпуска, компания, стоимость
6.	Модель автомобиля, марка, тип автомобиля, стоимость, год выхода, страна
7.	Планета, порядковый номер, расстояние до Солнца, происхождение названия, количество спутников, длительность суток

8.	Созвездие, расстояние от Земли, количество звёзд, самая яркая звезда, название сверхскопления

3. Создайте таблицу в HTML-файле. Загрузите текстовый файл через PHP и заполните созданную таблицу строками из файла.

ОТЧЁТ.

1. Для сдачи лабораторной работы необходимо оформить отчёт в виде файла в формате DOC/DOCX.
2. Указать задание к лабораторной работе.
3. Вывести исходный код.
4. Сделать скриншоты, приложить их в отчёт и дать описание.

Лабораторная работа №3.

Разработка PHP-страницы для взаимодействия с базой данных.

ЗАДАНИЕ.

4. Создать с помощью OpenServer базу данных PostgreSQL и пользователя с паролем для подключения.
5. Создать в этой базе данных таблицу, структура которой будет копировать структуру файла из лабораторной работы №2 (например, если выводятся страны и их столицы, то в качестве столбцов (полей) должны быть: страна, столица, год основания, язык и т.д.).
6. Заполнить созданную таблицу строками из файла и добавить новые (около 25-30 разных записей).
7. Реализовать на PHP подключение к созданной базе данных.
8. Считать и вывести на странице содержимое таблицы с данными.
9. Над выведенной таблицей на странице добавить поиск. Для этого создать форму (<form></form>), которая содержит:
   - выпадающие списки (<select></select>) по каждому из полей таблицы;
   - дополнительное текстовое поле (<input type=”text”>) для поиска по введенному тексту;
   - кнопку поиска (<button type=”submit”></button>).
10. При нажатии на кнопку поиска страница обновляется и выводятся результаты, удовлетворяющие заданным критериям поиска.

ОТЧЁТ.

5. Для сдачи лабораторной работы необходимо оформить отчёт в виде файла в формате

DOC/DOCX.

6. Указать задание к лабораторной работе.
7. Вывести исходный код.
8. Сделать скриншоты, приложить их в отчёт и дать описание.

Лабораторная работа №4.

Создание динамической страницы с использованием JavaScript.

ЗАДАНИЕ

1. Для таблицы сделать шапку (если нет) при помощи тегов <thead> и <th>.
2. У каждой ячейки шапки добавить «стрелки», которые отображают порядок сортировки по этому столбцу.
3. По нажатию на стрелку JavaScript должен поменять все строки таблицы таким образом, чтобы удовлетворяло выбранному порядку. Выбранная стрелка должна каким-либо образом выделиться как активная и поменять направление.
4. Добавить поле поиска <input type=”text”>, при вводе текста в которое в таблице скрываются строки, не удовлетворяющие введенному тексту. Поиск ведется по первому столбцу. Таблица перерисовывается при изменении (вводе с клавиатуры или стирании) любого символа в поле поиска.

РЕКОМЕНДАЦИИ К РАБОТЕ:

- при выполнении всех заданий в работе используется «чистый» JavaScript без дополнительных библиотек и фреймворков;
- за основу взять результаты лабораторной работы №3;
- все JS-скрипты должны быть выведены в отдельный файл, а в событиях вызываться соответствующие им функции, реализованные в этом файле.

МАТЕРИАЛЫ ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ:

- Структура таблицы должна быть реализована по спецификации HTML5:

```
<table>
<thead>
  <th>Столбец 1</th>
  <th>Столбец 2</th>
  …
</thead>
<tbody>
  <tr>
    <td>Ячейка 1</td>
    <td>Ячейка 2</td>
    …
  </tr>
  <tr>
    <td>Ячейка 3</td>
    <td>Ячейка 4</td>
    …
  </tr>
  …
</tbody>
</table>
```

- Для добавления стрелок можно использовать как рисунки, так и обычные символы. Для добавления события нажатия левой клавиши мыши на стрелку использовать метод: onClick=”функция_для_обработки();”;
- Для перемещения строк таблицы можно использовать различные варианты:
  - создание новой отсортированной таблицы и замена старой;
  - использование метода Node.insertBefore();
  - какие-либо другие (без использования дополнительных библиотек!);
- для скрытия строки таблицы использовать стили: element.style.display = ‘none’;

ОТЧЁТ.

1. Для сдачи лабораторной работы необходимо оформить отчёт в виде файла в формате DOC/DOCX.
2. Указать задание к лабораторной работе.
3. Вывести исходный код.
4. Сделать скриншоты, приложить их в отчёт и дать описание.

Лабораторная работа №5.

Фреймворк jQuery.

ЗАДАНИЕ

1. Все функции, реализованные в лабораторной работе №4, реализовать с помощью библиотеки jQuery.
2. Дополнительно в поле поиска реализовать под ним всплывающее меню, в котором можно выбрать доступные варианты (которые берутся из первого столбца таблицы). В качестве примера можно взять поисковое окно Google или Yandex. При выборе доступного варианта он должен автоматически добавиться в поле поиска и таблица автоматически перерисоваться (скрыть неподходящие строки).
3. С помощью jQuery UI добавить возможность изменения размеров таблицы с помощью метода resizable() и перетаскивание строк таблицы с помощью метода sortable().

МАТЕРИАЛЫ ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ:

- для создания всплывающего меню использовать элемент с привязкой к полю ввода (<input>), меню должно показываться, если курсор располагается в поле ввода, и скрываться, если курсор уходит из поля ввода;
- все события вида onClick=”функция();” и аналогичные убрать из html-файла и реализовать в js-файле с использованием jQuery(document).ready() и jQuery(элемент).click() и т.д.

ОТЧЁТ.

1. Для сдачи лабораторной работы необходимо оформить отчёт в виде файла в формате DOC/DOCX.
2. Указать задание к лабораторной работе.
3. Вывести исходный код.
4. Сделать скриншоты, приложить их в отчёт и дать описание.

Лабораторная работа №6.

Технология Ajax.

ЗАДАНИЕ

Реализовать онлайн-чат с возможностью:

- сохранения сообщений в базу данных;
- ввода имени пользователя при открытии окна чата;
- наличие блока для отображения всех сообщений чата из базы данных;
- наличие формы для ввода сообщения и его отправки на сервер с использованием технологии AJAX;
- считывание сервера на наличие новых сообщений при помощи технологии AJAX с частотой в 5-10 секунд и отображение новых сообщений.

ТРЕБОВАНИЯ:

- всё взаимодействие с сервером должно быть реализовано при помощи библиотеки jQuery и технологии AJAX без перезагрузки страницы.

ОТЧЁТ.

1. Для сдачи лабораторной работы необходимо оформить отчёт в виде файла в формате DOC/DOCX.
2. Указать задание к лабораторной работе.
3. Вывести исходный код.
4. Сделать скриншоты, приложить их в отчёт и дать описание.

Лабораторная работа №7.

Работа с системой управления контентом Joomla.

ЗАДАНИЕ

1. Установить и настроить систему управления содержимом Joomla 3.
2. Выбрать шаблон оформления.
3. Создать страницы и меню, которое должно содержать:
   - Страницу с информацией о себе (ФИО, фотография, группа, другая информация по желанию);
   - Страницу с ссылками на все предыдущие лабораторные работы;
   - Гостевую книгу;
   - Страницу с контактами (с возможностью отправки по e-mail);
   - Возможность «репоста» в социальные сети (как пример, социальные кнопки Яндекса);

ОТЧЁТ.

1. Для сдачи лабораторной работы необходимо оформить отчёт в виде файла в формате DOC/DOCX.
2. Указать задание к лабораторной работе.
3. Вывести исходный код.
4. Сделать скриншоты, приложить их в отчёт и дать описание.

Лабораторная работа №8.

Основы работы и анимация в Canvas

ЗАДАНИЕ

1. Создать html-страницу с Canvas-элементом на ней.
2. Подключить javaScript-файл, в котором в дальнейшем будет реализована логика рисования фигур.

3. Выбрать из таблицы вариант лабораторной работы (не более 2 человек на подгруппу).

Номер	Название	Объекты
1.	Игра «Крестики-нолики»	Крестик, нолик, игровое поле 3x3 клетки
2.	Игра «Пятнашки»	блоки с числами внутри (от 1 до 15), игровое поле
3.	Игра «2048»	блоки с числами внутри (от 2 до 1024, степени 2-ки), игровое поле 4x4 клетки
4.	Игра «Сапёр»	игровое поле, числа от 1 до 8, флажок, мина
5.	Игра «Змейка»	игровое поле, змейка (ломаная линия), квадрат (новая точка для увеличения длины змейки)
6.	Игра «Тетрис»	фигуры тетриса, игровое поле
7.	Игра «Танчики»	фигура «танчика», фигуры препятствий
8.	Игра «Шашки»	чёрные и белые круглые фигуры, игровое поле

4. Отобразить все объекты из выбранного варианта в статичном виде.

МАТЕРИАЛЫ ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ:

1. Создайте папку своего проекта.
2. В этой папке создайте файлы:
   - index.html
   - main.css
   - main.js
3. В файле index.html добавьте элемент CANVAS и с помощью CSS-стилей задайте его оформление. Чтобы создать Canvas-контекст, достаточно просто добавить элемент <canvas> в HTML-документ:

```
21:  <canvas id="myCanvas" width="300" height="150">
22:  Альтернативное содержимое, которое будет показано, если браузер не поддерживает Canvas.
23:  </canvas>
```

4. Чтобы нарисовать окружность, нужно выполнить такой код:

```
context.beginPath();
context.arc(75, 75, 10, 0, Math.PI*2, true);
context.closePath();
context.fill(); // Если нужен круг, можно залить окружность
```

5. Контуры Canvas позволяют рисовать фигуры любой формы. Сначала нужно нарисовать "каркас", а потом можно использовать стили линий или заливки, если это необходимо. Чтобы начать рисование контура, используется метод beginPath(), потом рисуется контур, который можно составить из линий, кривых и других примитивов. Как только рисование фигуры окончено, можно вызвать методы назначения стиля линий и заливки, и только потом вызвать функцию closePath() для завершения рисования фигуры.
6. Метод drawImage позволяет вставлять другие изображения (img и canvas) на канву. В браузере Opera также существует возможность рисования SVG-изображений внутри элемента canvas. drawImage довольно сложный метод, который может принимать три, пять или девять аргументов:
   - Три аргумента: Базовое использование метода drawImage включает один аргумент для указания изображения, которое необходимо вывести на канве, и два аргумента для задания координат.
   - Пять аргументов: Используются предыдущие три аргумента и еще два, задающие ширину и высоту вставляемого изображения (в случае если вы хотите изменить размеры изображения при вставке).
   - Девять аргументов: Используются предыдущие пять аргументов и еще четыре: два для координат области внутри исходного изображения и два для ширины и высоты области внутри исходного изображения для обрезки изображения перед вставкой в Canvas.
7. Используя метод fillRect, вы можете нарисовать прямоугольник с заливкой. С помощью метода strokeRect вы можете нарисовать прямоугольник только с границами, без заливки. Если нужно очистить некоторую часть канвы, вы можете использовать метод clearRect. Три этих метода используют одинаковый набор аргументов: x, y, width, height. Первые два аргумента задают координаты (x,y), а следующие два — ширину и высоту прямоугольника.

ОТЧЁТ.

1. Для сдачи лабораторной работы необходимо оформить отчёт в виде файла в формате DOC/DOCX.
2. Указать задание к лабораторной работе.
3. Вывести исходный код.

4. Сделать скриншоты, приложить их в отчёт и дать описание.

Лабораторная работа №9.

Создание изображения с помощью SVG-графики.

ЗАДАНИЕ

1. Научиться рисовать основные фигуры: прямоугольник, окружность, радугу, солнце.
2. Нарисовать дом, солнце, деревья и железную дорогу.

МАТЕРИАЛЫ ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ:

Перед тем как вы сможете что-нибудь рисовать, вам надо создать SVG-элемент. Думайте об SVG-элементе, как о холсте, на котором отрисовываются все ваши визуальные образы(В такой трактовке, SVG концептуально схож с элементом HTML - canvas). Как минимум, хорошо задать ширину и высоту с помощью атрибутов width и height, соответственно. Если вы их не зададите, SVG растянется на всю площадь блока.

1. Создайте папку своего проекта.
2. В этой папке создайте файл *index.html*
3. В код HTML этого файла добавьте элемент SVG.

```
<svg width="500" height="50"></svg>
```

4. Для рисования элементов изображения используйте теги *<rect></rect>* (прямоугольник), *<circle></circle>* (окружность), *<line></line>* (линия), *<ellipse></ ellipse >* (эллипс).
5. Для каждой из фигур задайте цвет заливки и контура с помощью атрибутов *fill* и *stroke*.

ОТЧЁТ.

1. Для сдачи лабораторной работы необходимо оформить отчёт в виде файла в формате DOC/DOCX.
2. Указать задание к лабораторной работе.
3. Вывести исходный код.
4. Сделать скриншоты, приложить их в отчёт и дать описание.

Лабораторная работа №10.

Основы Python

ЗАДАНИЕ

Имеется список студентов, который представлен в виде структуры [[№, ФИО, Возраст, Группа],[№, ФИО, Возраст, Группа],[№, ФИО, Возраст, Группа]]. Преобразуйте список в словарь вида: {№: [ФИО, Возраст, Группа], №: [....], №: [....]}
Добавьте для словаря возможность (без преобразования словаря обратно в список):

- Вариант 1. Увеличить возраст конкретного студента на 1. Поиск по «ФИО» («ФИО» студента необходимо ввести с клавиатуры).
- Вариант 2. Изменить «ФИО» студента. Поиск по «ФИО» (старое и новое «ФИО» студента необходимо ввести с клавиатуры).
- Вариант 3. Увеличить возраст конкретного студента на 1. Поиск по «№» («№» студента необходимо ввести с клавиатуры).
- Вариант 4. Изменить группу студента. Поиск по «ФИО» («ФИО» студента и новый номер группы необходимо ввести с клавиатуры).
- Вариант 5. Удалить запись о студенте. Поиск по «№» («№» студента, которого нужно удалить из списка, задается с клавиатуры)
- Вариант 6. Если возраст студента больше 22 уменьшить его на 1.
- Вариант 7. Если возраст студента равен 23, удалить его из списка.
- Вариант 8. У всех студентов с фамилией «Иванов» увеличить возраст на 1.
- Вариант 9. У студентов с фамилией «Иванов» изменить фамилию на «Сидоров».
- Вариант 10. Поменять «ФИО» и «Группа» местами.

МАТЕРИАЛЫ ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ:

1. Установить Python версии 3.
2. Создать файл с расширением *.py* (например, *lab.py*)
3. В созданном файле с помощью любого текстового редактора написать исходный код, выполняющий поставленные в задании функции.
4. Запустить исходный код командой и удостовериться в правильности работы программы.

```
1: python lab.py
```

ОТЧЁТ.

1. Для сдачи лабораторной работы необходимо оформить отчёт в виде файла в формате DOC/DOCX.
2. Указать задание к лабораторной работе.
3. Вывести исходный код.

4. Сделать скриншоты, приложить их в отчёт и дать описание.

Лабораторная работа №11.

Создание Web-сервера на Python

ЗАДАНИЕ

Установка и настройка web-сервера на Python и размещение на нём личного сайта разработчика с общей информацией.

Дополнительная возможность: гостевая книга из базы данных. Использование технологии AJAX для отправки и отображения новых сообщений.

МАТЕРИАЛЫ ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ:

1. Установить библиотеки Web.py и SQLAlchemy для Python
2. Установить и настроить СУБД PostgreSQL
3. Создать базу данных, хранящую всю необходимую информацию о сайте
4. Создать основной файл web-сервера (например, *server.py*)
5. С помощью текстового редактора написать исходный файл для запуска web-сервера и подключения к СУБД PostgreSQL с помощью библиотеки SQLAlchemy, а также HTML-код для отображения страниц сайта.

ОТЧЁТ.

1. Для сдачи лабораторной работы необходимо оформить отчёт в виде файла в формате DOC/DOCX.
2. Указать задание к лабораторной работе.
3. Вывести исходный код.
4. Сделать скриншоты, приложить их в отчёт и дать описание.

Литература

1. Теория информации : учеб.-справ. пособие / А. А. Смагин. - Ульяновск : УлГУ, 2007. - 103 с. - Библиогр.: с. 102. - б/п.
2. PHP7 для начинающих с пошаговыми инструкциями : пер. с англ. / МакГрат Майк. - Москва : Э, 2018. - 253 с.
3. PHP на примерах / Е. В. Поляков. - Санкт-Петербург : Наука и техника, 2017. - 256 с.
4. Основы WEB-технологий : учеб.-метод. пособие по курсу "Программирование для Internet" / Т. В. Бажанова, Е. В. Филаткина; УлГУ, ФМИиАТ. - Ульяновск : УлГУ, 2016. - 72 с. - б/п.
5. Теория информации : методические указания к выполнению курсовых работ / А. А. Смагин, А. А. Булаев; Ульяновск. гос. ун-т. - Ульяновск : УлГУ, 2018. - 32 с. - Библиогр.: с. 32. - б/п.
6. Зудилова Т.В., Буркова М.Л. Web-программирование: HTML: Учебное пособие. - СПб.: НИУ ИТМО, 2012. - 70 с.
7. Основы HTML [Электронный ресурс] // HTMLBOOK.RU: Для тех, кто делает сайты. URL: http://htmlbook.ru/content/osnovy-html/

ПРИЛОЖЕНИЯ

Приложение 1. Титульный лист к отчёту по лабораторной работе

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Математики, информационных и авиационных технологий

Кафедра Телекоммуникационные технологии и сети

ЛАБОРАТОРНАЯ РАБОТА №___

(название темы)

ПО ДИСЦИПЛИНЕ

(название дисциплины)

по направлению *бакалавриата "Инфокоммуникационные технологии и системы связи"*

по направлению *бакалавриата " Информационные системы и технологии"*

по направлению *магистратуры "Инфокоммуникационные технологии и системы связи"*

Работу выполнил студент _____ _____ _____
группа подпись, дата Ф.И.О.

Преподаватель _____ _____ _____
должность подпись, дата Ф.И.О.

Оценка: _____

У Л Ь Я Н О В С К
2019