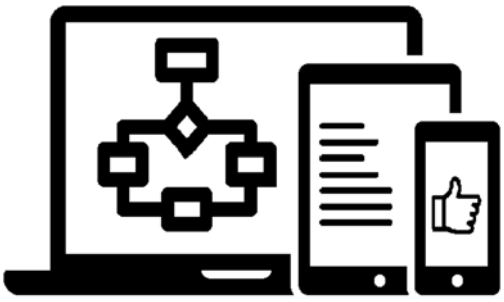
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

ОСНОВЫ WEB-ПРОГРАММИРОВАНИЯ

Методические рекомендации к лабораторным работам для студентов направлений подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» очной формы обучения



Могилев 2021

УДК 004.43
ББК 32.973.26–018.1
О75

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «18» февраля 2021 г., протокол № 7

Составитель канд. техн. наук, доц. Э. И. Ясюкович

Рецензент канд. техн. наук, доц. В. М. Ковальчук

Методические рекомендации содержат основные базовые теоретические сведения, некоторые приемы реализации задач, а также практические задания для выполнения лабораторных работ по всем темам курса.

Учебно-методическое издание

ОСНОВЫ WEB-ПРОГРАММИРОВАНИЯ

Ответственный за выпуск	В. В. Кутузов
Корректор	А. А. Подошевко
Компьютерная верстка	Е. В. Ковалевская

Подписано в печать 18.05.2021 . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс. Печать трафаретная. Усл. печ. л. 2,79 . Уч.-изд. л. 3,0 . Тираж 26 экз. Заказ № 361.

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев

Содержание

Введение

Целью изучения дисциплины «Основы web-программирования» является приобретение специальных знаний, умений и практических навыков, необходимых менеджеру по информационным технологиям для разработки интернет-приложений.

Язык html (Hyper Text Markup Language) используется для добавления разметки в обычный текст, позволяет создавать статические и динамические сайты и является языком, описывающим структуру и семантику web-документа.

Стандарт языка html непрерывно обновляется и почти каждый год выходит его новая версия. Версия html 2.0 была опубликована в 1995 г., html 4.01 – основная версия html – в конце 1999 г. В настоящее время наиболее популярна версия html -5, являющаяся расширением html 4.01.

Каскадные таблицы стилей (Cascading Style Sheets – CSS) влияют на отображение страниц в окнах браузеров (цвета, шрифты, фоновые изображения, интервалы между строками, отступы, границы, эффекты и даже анимация элементов). Благодаря CSS можно производить изменения, относящиеся ко всем страницам сайта, редактируя при этом лишь один единственный файл таблицы стилей.

Для упрощения процесса разработки JavaScript скриптов Джоном Резигом была разработана библиотека jQuery, которая постоянно дополняется добровольцами. Основная цель создания jQuery – возможность создания многоразовых фрагментов кода JavaScript, позволяющих упростить процесс использования их в html-документах. Также библиотека jQuery предоставляет удобный программный интерфейс приложения API для работы с AJAX – интерактивным пользовательским интерфейсом.

Методические рекомендации предназначены для изучения основ технологии интернет-программирования и состоят из двух частей. Первая часть содержит 11 лабораторных работ, вторая – шесть. В каждой части излагается краткий теоретический материал по вопросам соответствующей темы, предлагаются технологии решения некоторых типовых задач, приводятся задания для выполнения работ, а также контрольные вопросы.

Лабораторная работа № 1 посвящена изучению языка ***html*** и CSS. Лабораторные работы № 2–11 ориентированы на изучение основ языка JavaScript и библиотеки jQuery, лабораторные работы № 12–17 – на изучение серверного языка PHP.

Каждая лабораторная работа рассчитана на 2 ч, а ее цель соответствует названию.

Выполнение лабораторной работы производится в следующем порядке:

1) ознакомиться с теоретическими положениями работы;

2) выбрать из таблиц «Варианты заданий», которые содержатся в конце каждой лабораторной работы, согласно варианту, указанному преподавателем, задания и исходные данные, выполнить задания и оформить отчет.

1 Лабораторная работа № 1. Создание web-страниц с использованием HTML и CSS

Большинство web-страниц содержат описание разметки гипертекста на языке HTML5, основным элементом которого является тег, представляющий собой правило, согласно которому информация отображается в окне браузера.

Для создания и редактирования web-страниц используются текстовые редакторы, такие как ***Notepad++***, ***Sublime Text3***, ***Geany***, ***Vim***, ***Emacs*** и др. Простейшая структура ***html***-страницы имеет вид:

```
<!DOCTYPE html>
<html>
  <head>
      <meta charset="utf-8"
      <title> </title>
  </head>
  <body>
      содержание html–страницы
  </body>
</html>
```

Первым разделом ***html***-страницы является раздел DOCTYPE, определяющий правила, содержащиеся в файле объявления типа документа.

Парный тег <***html***> </***html***> описывает корневой элемент ***html***-страницы, информирует браузер о том, что внутри него содержится код web-страницы, содержащий два раздела: ***head*** и ***body***. Раздел ***head*** содержит информацию о способе представления данных и стилях, о исполняемых скриптах, а также данные для поисковых систем. Раздел ***body*** – это тело документа, содержащее теги ***html***-страницы

В HTML5 присутствие тегов <***html***>, <***head***> и <***body***> не является обязательным.

Язык HTML5 содержит примерно 100 различных тегов, предоставляющих широкий спектр возможностей для работы с текстом и мультимедиа.

Теги могут содержать дополнительные свойства, называемые атрибутами – параметрами, влияющими на их работу. Параметры тегов перечисляются после имени тега через пробел с определенными значениями, записываемыми в кавычках:

<***тег атрибут$_1$="значение" атрибут$_2$="значение" ...***>

Атрибуты могут записываться в любом порядке.

У тегов есть собственные атрибуты, но существуют также глобальные атрибуты, доступные всем тегам, например, ***style***, ***id***, ***class***. Так, тег <***body***> обладает рядом атрибутов, задающих глобальные параметры фона, отступов, текста и гиперссылок.

В HTML5 реализовано немало новых тегов, позволяющих повысить эффективность обработки сайта поисковыми системами, например, тегов контейнеров <***header***>, <***footer***>, <***aside***> и <***nav***>.

Теги <***header***> и <***footer***> предоставляют область верхнего и нижнего информационного блока. В теге <***header***> располагаются логотипы, основные

разделы сайта, элементы управления, поиска, регистрации и т. д., а в теге <***footer***> может содержаться контактная информация.

Тег <***aside***> удобно использовать для оформления блока меню, боковой панели или поясняющего раздела, а тег <***nav***> – навигации по сайту.

Часто содержимое сайта можно рассматривать как набор статей похожей структуры и оформления. Для таких целей удобно использовать теги <***main***>, <***article***> и <***section***>.

Для создания и изменения стилей элементов web-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, используются каскадные таблицы стилей CSS.

Основы синтаксиса CSS. Если в HTML ключевым элементом являлся тег, то в CSS таким элементом является селектор, представляющий собой некоторое имя стиля, для которого добавляются параметры форматирования. В качестве селектора могут выступать теги, атрибуты тегов, классы и идентификаторы.

Теги-селекторы позволяют определять оформление тегов, а селекторы атрибутов используются для задания их стилей. Классы и идентификаторы повышают гибкость использования стилей и делают стиль независимым от конкретного тега.

Общий способ записи селектора, на примере тега <***h1***>, имеет вид: селектор (***h1***) – свойство (***background***) – значение свойства ***(#EEE***).

```
h1 {
   background: #EEE;
   …
}
```

Селектор сообщает браузеру, какой элемент необходимо форматировать, а блок объявления (код в фигурных скобках), содержит форматирующие команды, указывающие свойства со значениями.

CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции, поэтому форма записи зависит от вкуса разработчика. Хорошим считается стиль, когда каждое свойство указывается на отдельной строке, но если в селекторе только один стиль, его часто записывают в одну строчку:

```
body { background: #F5E2D0; }
```

В CSS3 можно выделить следующие виды стилей, определяющие способ их подключения к основному документу:

– встроенные;

– внутренние;

– связанные.

Каждый из этих стилей имеет определенные преимущества в плане гибкости использования и эффективности загрузки страницы.

В работе предлагается, используя язык разметки гипертекста ***html5*** и каскадную таблицу стилей CSS3, разработать web-страницу формирования меню, содержащего главную ленту с графическим элементом и два пункта с выпадающими подменю (рисунок 1.1). Пример задачи приведен в листингах 1 и 2.

Рисунок 1.1 – Вид меню

Листинг 1 – html-код страницы

```
<!DOCTYPE html>
<html>
<head>
<title>Меню</title>
<link rel="stylesheet" type="text/css"
href="css/style.css" />
</head>
<body>
<div id="mainmenu">
   <ul id="nav">
<li id="settings">
<a href="#"><img
src="images/settings.png"></a>
</li>
  <li><a href="">Пункт-1</a>
    <ul>
      <li><a href="#">Команда 1-
1</a></li>
      <li><a href="#">Команда 1-
2</a></li>
    </ul>
  </li>
 </ul>
</div>
</body>
</html>
```

Листинг 2 – CSS-код страницы

```
#mainmenu    {
  float:left;   }
#mainmenu ul {
  margin: 10; /*   отступ   ленты   меню
сверху*/
  padding: 5;          /* отступ ленты меню
слева*/
  list-style: none;     }
#mainmenu ul li      {
  position: relative;
  float:left;   }
#mainmenu ul li ul, #mainmenu ul li ul li  {
  width:130px;
  color:red;   /*      */                }
#mainmenu li ul      {
  position: absolute;
  left: 0;
  top: 29px;
  display: none;
  float:left;   }
#mainmenu ul li a    {
  float:left;
  color:blue;  /*Цвет текста глав меню  */
  width:80px; /*Длина ленты глав меню  */
  font-size:16px;
  padding: 10px 0 10px 0;
  text-align:center;
  background:   #00f0f0;/*цвет   фона   гл
меню*/
}
#mainmenu li ul li a          {
  padding:5px 0 3px 10px;
  text-align:left;   font-size:12px;
  width:80px;
  background: #EEEEEE;  /*Цвет      поля
группы выбр-го подменю*/    }
#mainmenu li ul li a:hover  {
  background:   #0ffff0;       /*Цвет   фона
подменю */
  color:blue; }
#settings a {          /*Графич-й эл-т */
  padding: 18px;
  height: 19px;       /*Высота   графич   эл-
та */
  font-size: 10px;   line-height: 24px;      }
* html #mainmenu  ul  li  {float:  left;  height:
1%;}
* html #mainmenu ul li a { height: 1%; }
#mainmenu li:hover ul,
#mainmenu  li.over ul { display: block;     }
```

Порядок выполнения работы.

Выполнить задания, приведенные в колонках 2–5 таблицы 1.1. В колонке 2 указаны темы, для которых необходимо разработать web-страницу формирования меню, в колонке 3 – параметры главной ленты меню: направление главной ленты меню (гориз – горизонтальное, верт – вертикальное); фон – цвет фона ленты меню; текст – цвет текста пунктов меню. В колонках 4 и 5 указано количество пунктов второго и третьего подменю (два или три), цвет фона подменю и цвет текста его пунктов.

Главное меню должно отстоять от верхней границы окна браузера на 20 ***px***, слева – на 12 ***px*** и содержать слева графический элемент.

Таблица 1.1 – Варианты заданий

Номер варианта	Тема	Главная лента меню: фон/текст	Подменю 1: пунктов/фон/текст	Подменю 2: пунктов/фон/текст
1	2	3	4	5
1	Торговая фирма	Гориз/Pink/Red	3/Blue/Blue	2/Aqua/Violet
2	Ремонт авто	Верт/Coral/Blue	2/Navy/Navy	3/Olive/Aqua
3	Расписание занятий	Гориз/Gold/Navy	3/Olive/Aqua	2/Navy/Navy
4	Летняя одежда	Верт/Khaki/Aqua	2/Aqua/Violet	3/Blue/Blue
5	Ремонт компьютеров	Гориз/Plum/Black	3/Black/Red	2/Tan/Gold
6	Швейное ателье	Верт/Purple/Violet	2/Tan/Gold	3/Violet/Blue
7	Ремонт телевизоров	Гориз/Indigo/Tan	3/Violet/Blue	2/Purple/Navy
8	Головные уборы	Верт/Peru/Coral	2/Purple/Navy	3/Blue/Red
9	Зимняя одежда	Гориз/Gray/Gold	3/Blue/Red	2/Olive/Aqua
10	Продукты питания	Верт/Red/Plum	2/Olive/Aqua	3/Peru/ Gold
11	Детские игрушки	Гориз/Lime/Peru	3/Peru/ Gold	2/Pink/ Blue
12	Модели авто	Верт/Aqua/Gray	2/Pink/ Blue	3/Blue/Blue

Контрольные вопросы

1 Для чего используются CSS в web-программировании?
2 Как построить гиперссылку, заголовок и список в html-документе?
3 Какова структура объявления стиля?
4 Из каких разделов состоит html-документ?
5 Какие программные средства используются для создания web-страниц?
6 Как изменить параметры текста в html-документе (размер, цвет, стиль)?
7 Как создать таблицу в html-документе и вставить рисунок в ее ячейку?
8 Как подключить таблицу CSS стилей к html-документу?
9 Как построить гиперссылку на графическом элементе?
10 Как установить параметры ленты меню web-страницы?

2 Лабораторная работа № 2. Создание простых скриптов на JavaScript

Для расширения функциональных возможностей web-страниц используются скрипты ***JavaScript***, которые включаются в html-документы несколькими способами:

– в теговом контейнере <***body***>...</***body***>:

```
<body>
…
<script > команды скрипта</script>
…
</body>
```

– в контейнере <***head***>...</***head***>, если скрипт представляет собой функцию, вызываемую в ответ на какое-либо событие:

```
<head>
…
<script type=”text/javascript”> команды сценария </script>
</head>
```

– во внешнем файле с расширением ***js:***

```
<head>
…
<script type = "text/javascript" src = "my.js"> </script>
</head>
```

Любые текстовые данные в языке ***JavaScript*** являются строками. В ***JavaScript*** не существует отдельного типа «символ», который есть в ряде других языков, а внутренний формат для строк – всегда UTF-16, вне зависимости от кодировки страницы.

Для создания строк в ***JavaScript*** можно использовать одинарные, двойные или обратные кавычки.

Одинарные и двойные кавычки работают одинаково, а строка в обратных кавычках может содержать произвольное выражение, заключенное в ${…}, и располагаться более чем в одной строке.

Многострочные строки также можно создавать с помощью одинарных и двойных кавычек, используя символ перевода строки ***\n***.

Свойство ***length*** содержит длину строки:

```
alert( `My\n`.length );        // 3
```

здесь ***\n*** – один спецсимвол, поэтому длина строки равна 3.

Получить символ, который занимает позицию ***pos***, можно с помощью квадратных скобок: [***pos***]. Также можно использовать метод ***charAt***.

Поиск подстроки. Метод ***str.indexOf(substr, pos)*** ищет подстроку ***substr*** в строке ***str***, начиная с позиции ***pos***, и возвращает позицию, на которой располагается совпадение, либо –1 при отсутствии совпадений. Чтобы найти все вхождения подстроки, можно использовать метод ***indexOf()*** в цикле.

Для получения подстроки можно использовать методы ***substring***, ***substr*** и ***slice***. Например, ***str.slice(start [, end])*** вернет часть строки от ***start*** до (не включая) ***end***.

Порядок выполнения работы.

Выполнить следующие задания.

1 Создать строку текста из 22 первых букв русского алфавита: var str = 'abcde … '. Используя функцию ***alert*** вывести символы с указанными в столбце «Задание 1» таблицы 2.1 номерами и разделить их номером варианта, например, для варианта 11: а-11-с-11-е11 и т. д.

2 Построить строку из цифр «Номера символов» первого задания. Из полученной строки, состоящей из 12 цифр, выделить четыре трехзначных числа. Используя полученные числа и заданные в колонке «Задание 2» таблицы 2.1 операции, построить оператор присваивания с построенным арифметическим выражением, полученный результат вывести в окно браузера.

3 Построить строку текста из букв, номера которых заданы в «Задание 1». Используя свойство ***innerHTML*** метода ***document.getElementById(id),*** вывести на страницу html построенную строку.

4 С помощью функции ***confirm*** построить запрос, содержащий две кнопки: Да и Нет. В зависимости от выбранной кнопки вычислить заданные в «Задании 3» таблицы 2.1 выражения: для ***Да*** – ***y***, для ***Нет*** – ***f***. Результат вывести на страницу ***html*** используя функцию ***writeln***.

Таблица 2.1 – Задания к лабораторной работе

Номер варианта	Задание 1 (номера символов)	Задание 2 (операции)			Задание 3 (арифметическое выражение)	
					$y =$	$f =$
1	01, 03, 05, 06, 08, 09	+	–	*	$(a + b)^2/c$	$d/(f - e/2)$
2	02, 03, 04, 06, 07, 15	+	–	/	(a * b) – c^2	$2*d/(f + e/4)$
3	04, 05, 07, 09, 12, 18	+	–	%	$(a - b/c)^2$	$d*(2*f - e/2)$
4	03, 05, 06, 12, 20, 21	/	*	–	$(a^2 + b)/c$	$d/(f/5 + e\text{^}2)$
5	01, 02, 08, 15, 17, 19	/	*	+	$(a - b/c)^2/c$	$2*d/(f/1{,}4 - e/2)$
6	13, 15, 16, 17, 18, 20	/	*	%	$(a + b/c^2)/c$	$d/(f*e/2) + d$
7	05, 07, 09, 13, 16, 18	*	+	–	$(a - b/a)/c^2$	$d/(f*e + 2*d) - d$
8	03, 04, 09, 12, 14, 17	*	+	/	$(a + b/a^2)/2*c$	$d/(f*e - 2*d) + e$
9	02, 09, 16, 17, 18, 21	*	+	%	$(a - b/a*2)/c^2$	$d/(e*f/2) + d*e/f$
10	03, 07, 08, 09, 10, 20	–	*	/	$(a + b/c^2)/c - a$	$d/(f*e/2 + d) - 2*f$
11	01, 03, 05, 07, 10, 11	–	*	+	(a – $b/c/2)/c^a$	$d/(f*e + 2*f) + d$
12	06, 08, 09, 12, 13, 19	–	*	%	$(a + b/c+2)/c–2*b$	$d/(f*e - 2/f) - d$

Контрольные вопросы

1 Какие функции вывода на страницу ***html*** Вы знаете?
2 Какие математические операции используются в ***JavaScripn***?
3 Как извлечь символ строки по его номеру?
4 Как встроить ***JavaScript***-код в ***html***-документ?
5 Как выделить символ из строки текста?
6 Какие комментарии используются в языке ***JavaScript***?
7 Для каких целей используются методы ***prompt*** и ***Confirm***?
8 Для чего используются методы ***document.write()***, ***alert()***, ***console.log()***?
9 Какие способы включения ***JavaScript***-кода в ***html***-документ Вы знаете?
10 Прокомментируйте технологию использования метода ***document.getElementById(id)*** вывода на страницу ***html***.

3 Лабораторная работа № 3. Изучение функций обработки событий JavaScript

Событие – это сигнал от браузера о том, что что-то произошло. Среди событий можно выделить: события мыши: ***click***, ***contextmenu***, ***mouseover*** / ***mouseout***, ***mousedown*** / ***mouseup***, ***mousemove***; события на элементах управления: ***submit***, ***focus***; клавиатурные события: ***keydown*** и ***keyup***.

Событию можно назначить обработчика, т. е. функцию, которая сработает, как только событие произошло.

Если при наступлении события требуется произвести много действий, то удобно написать сценарий в виде функции и разместить его в контейнере <***script***> …<***/script***>, предназначенном для сценариев. Например, для вывода модуля заданного числа используется функция ***Math.abs***, а для округления чисел – функции ***Math.round***, ***Math.ceil*** и ***Math.floor***, а также методы ***toFixed*** и ***toPrecision***.

Функции ***Math.min***, ***Math.max***, ***Math.sqrt***, ***Math.pow***, ***Math.random*** используются для определения минимального, максимального значений, вычисления квадратного корня, возведения в степень и генерации псевдослучайных чисел с равномерным законом распределения.

Для работы со строками текста используются следующие методы: ***length***, ***toUpperCase***, ***toLowerCase***, ***substr***, ***substring***, ***slice***, ***indexOf***, ***replace***, ***split*** и функция ***join***. Чтобы обратить внимание пользователя web-сайта на определённый элемент ***html***-документа, его свойства можно менять, например, цвет или размер, при попадании на него курсора мышки, а при снятии курсора восстанавливать прежние значения.

Для взаимодействия с пользователем в ***JavaScript*** определен механизм событий. Например, когда пользователь нажимает кнопку, то возникает событие нажатия кнопки. В коде ***JavaScript*** можно определить возникновение события и обработать его.

Событие в ***JavaScript*** – это определенное действие, которые вызвано либо пользователем, либо браузером, например, клик мыши по кнопке, движение мыши, наведение фокуса на элемент, изменение значения в каком-либо текстовом поле, изменение размеров окна браузера и т. д.

В ***JavaScript*** используются следующие типы событий:

- события мыши (перемещение курсора, нажатие мыши и т. д.);
- события клавиатуры (нажатие или отпускание клавиши клавиатуры);
- события жизненного цикла элементов (например, загрузки web-страницы);
- события элементов форм (нажатие кнопки на форме и т. д.);
- события, возникающие при изменении элементов DOM;
- события, возникающие при касании на сенсорных экранах;
- события, возникающие при возникновении ошибок.

В таблице 3.1 приведены события ***JavaScript***, объекты, которые могут их генерировать, и причины возникновения соответствующих событий.

Таблица 3.1 – События ***JavaScript***

Событие	Объект	Причина возникновения
Abort	Image	Прерывание загрузки изображения
Blur	Button, Checkbox, FileUpload, Frame, Layer, Password, Radio, Reset, Select, Submit, Text, Textarea, Window	Потеря фокуса элемента
Change	FileUpload, Select, Text, Textarea	Смена значения
Click	Area, Button, Checkbox, Document, Link, Radio, Reset, Submit	Клик мыши на элементе
DblClick	Area, Document, Link	Двойной клик на элементе
DragDrop	Window	Перемещение в окно браузера
Focus	Button, Checkbox, FileUpload, Frame, Layer, Password, Radio, Reset, Select, Submit, Text, Textarea, Window	Установка фокуса на элементе
KeyDown	Document, Image, Link, Textarea	Нажатие клавиши на клавиатуре
KeyPress	Document, Image, Link, Textarea	Удержание клавиши на клавиатуре
KeyUp	Document, Image, Link, Textarea	Отпуск клавиши
Load	Document, Image, Layer, Window	Загрузка элемента
MouseDown	Button, Document, Link	Нажатие кнопки мыши
MouseMove	Window	Мышь в движении
MouseOut	Area, Layer, Link	Мышь выходит за границы элемента
MouseOver	Area, Layer, Link	Мышь находится над элементом
MouseUp	Button, Document, Link	Отпускание кнопки мыши
Move	Frame	Перемещение элемента
Reset	Form	Сброс формы
Resize	Frame, Window	Изменение размеров
Select	Text, Textarea	Выделение текста
Submit	Form	Передача данных
Unload	Window	Выгрузка текущей страницы

Для использования событий используются их обработчики, которые определяют, что будет происходить при возникновении определённого события. Обработчики событий в JavaScript имеют вид: ***onНазваниеСобытия***, т. е. вначале записывается приставка «***on***», а за ней – название события. Например, обработчики событий: ***onFocus***, ***onClick***, ***onSubmit*** и др. Область применения событий в ***JavaScript*** огромна.

Рассмотрим простейшую обработку событий. Например, на web-странице записан элемент div:

```
<div id="rect" onclick="alert('Нажато')"
        style="width:50px;height:50px;background-color:blue;"></div>
```

Здесь определен обычный блок ***div*** с атрибутом ***onclick***, который задает обработчик события нажатием на блок ***div***. То есть, чтобы обработать какое-либо событие, нам необходимо определить для него обработчик, который представляет собой соответствующий код на языке ***JavaScript***.

Также можно вынести все действия по обработке события в отдельную функцию:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
  </head>
  <body>
  <div id="rect" onclick="displayMessage()"
        style="width:50px;height:50px;background-color:blue;"></div>
        <script>
              function displayMessage(){

                 alert('Нажато');
              }
        </script>
  </body>
</html>
```

Теперь обработчиком события будет выступать функция ***displayMessage***.

Обработчик событий может быть назначен прямо в HTML-разметке, в атрибуте, который называется ***on*** <событие>. Например, чтобы назначить обработчик события ***click*** на элементе ***input***, можно использовать атрибут ***onclick***:

```
<input value="Нажми меня" onclick="alert('Клик!')" type="button">
```

При клике мышкой на кнопке выполнится код, указанный в атрибуте ***onclick***.

Порядок выполнения работы.

Выполнить задания, приведенные в таблице 3.2.

Таблица 3.2 – Задания к лабораторной работе

Номер варианта	Задание 1		Задание 2
	Используя Math.abs, вычислить модуль числа	Округлить число до *n* знаков в дробной части, используя функцию	Выполнить операции с элементами одномерного массива из *n* вещественных чисел, используя функцию
1	278,785	Math.round, *n* = 1	Math.min, *n* = 8
2	547,473	Math.ceil, *n* = 2	Math.max, *n* = 9
3	182,487	Math.floor, *n* = 3	Math.sqrt, *n* = 5 и просуммировать
4	264,289	toFixed, *n* = 2	Math.pow, *n* = 6 и просуммировать
5	452,297	Math.round, *n* = 2	Math.random, *n* = 9 и просуммировать
6	984,573	Math.ceil, *n* = 4	Math.min, *n* = 7
7	893,284	Math.floor, *n* = 1	Math.max, *n* = 8
8	487,651	toFixed, *n* = 3	Math.sqrt, *n* = 5 и просуммировать
9	774,652	Math.round, *n* = 3	Math.pow, *n* = 9 и просуммировать
10	682,571	Math.ceil, *n* = 1	Math.random, *n* = 7 и просуммировать
11	455,628	Math.floor, *n* = 2	Math.min, *n* = 9
12	671,475	toFixed, *n* = 4	Math.max, *n* = 6

Контрольные вопросы

1 Какие функции и методы округления чисел Вы знаете?

2 Прокомментируйте технологию использования функций ***Math.sqrt, Math.pow, Math.random***.

3 Прокомментируйте технологию использования функций ***isNaN***, ***isFinite***, ***parseInt***, ***parseFloat***.

4 Какая функция используется для округления вещественного числа?

5 Как получить модуль числа?

6 В каких ситуациях возникает необходимость использования регулярных выражений при работе с текстом?

7 Как сформировать последовательность псевдослучайных чисел с равномерным распределением?

8 Как выполняется глобальный поиск и замена символа в строке текста?

9 Для чего используется метод ***indexOf*** при работе со строками текста?

10 Какие методы для работы со строками Вы знаете?

4 Лабораторная работа № 4. Изучение операторов ветвлений и циклов JavaScript

Операторы ветвления используются для организации выполнения блоков кода при выполнении или невыполнении некоторых условий. К таким операторам относятся конструкции ***if***, ***else***, ***switch***.

Синтаксис конструкции ***if***:

```
if (логическое выражение) {
        код если логическое выражение true
} else {
        код если логическое выражение false
}
```

Синтаксис конструкции ***switch***:

```
switch (переменная) {
  case '1':
     код, выполняемый, если переменная имеет значение 1;
  break;
  ...
  case 'n':
     код, выполняемый, если переменная имеет значение n;
  break;
  default:
     код, выполняемый, если переменная не совпала ни с одним значением;
  break;
}
```

Синтаксис конструкции ***for***:

```
for (начальное значение; условие окончания цикла; ) изменение значения{
     тело цикла
}
```

Примеры

```
1 Вывести четные числа от 10 до 0
for (var i = 10; i >= 2; i -= 2) {
        document.write(i + ' ');
}
```

```
2 Вывести элементы массива
var books = ['JavaScript', 'PHP', html, 'CSS'];
for (var i = 0; i < books.length; i++) {
        var str = (i + 1) + '. ' + books[i] + '<br>';
        document.write(str);
}
```

Синтаксис конструкции while:

```
while ( пока выражение истинно ) {
     выполнять код
}
```

Пример 1 – Вывести числа от 1 до 10:

```
var i = 1;
while (i <= 10) {
     document.write(i + ' ');
     i++;
}
```

Инструкция ***break*** используется для принудительного выхода из цикла, а инструкция ***continue*** – для перехода к следующей итерации цикла.

Пример 2 – Вывести четные числа от 2 до 10:

```
var result = ' ';
for (var i = 2; i <= 10; i++) {
    if (i % 2) continue;
    result += i + ' ';
}
document.write(result);
```

Порядок выполнения работы.

Разработать консольное приложение на языке ***JavaScript*** для решения следующих задач.

1 Ввести переменную ***lang***, которая может принимать значения: рус, анг, бел или нем. В переменной *msw* сформировать массив дней недели на русском, английском, белорусском или немецком языке в зависимости от варианта. Задачу решить с помощью оператора ***if***; ***switch-case*** или многомерного массива без ***if*** и ***switch-case*** в зависимости от варианта.

2 Дана строка вида 'ab12cde345'. Проверить, является ли символ с заданным номером (*k*) этой строки буквой, а сумма ее цифр – четной.

3 Ввести дату и по ней определить: ВГ – время года (зима, весна, лето, осень); ДМ – декаду месяца; МГ – месяц; ВсГ – високосный / не високосный год.

4 Ввести массив из 25 целых вещественных чисел и определить: СЧЭ – сумма четных элементов; ПЭНН – произведение элементов с нечетными номерами; СЭКЗ – сумма элементов, номера которых кратны трем; ПНЧЭ – произведение нечетных элементов массива.

Варианты задач приведены в таблице 4.1.

Таблица 4.1 – Варианты задач

Номер варианта	Задача 1		Задача 2		Задача 3	Задача 4
	lang	операт/ массив	строка	*k*	определить	определить
1	рус	If	'abcde12345'	2	ВГ; ДМ	СЧЭ; ПЭНН
2	рус	switch-case	'ab123cde45'	4	ВГ; МГ	СЧЭ; СЭКЗ
3	рус	массив	'abcd12345e'	6	ВГ; ВсГ	СЧЭ; ПНЧЭ
4	анг	If	'ab12cde345'	8	ДМ; ВГ	ПЭНН; СЧЭ
5	анг	switch-case	'a12bcde345'	7	ДМ; МГ	ПЭНН; СЭКЗ
6	анг	массив	'a1234bcde5'	1	ДМ; ВсГ	ПЭНН; ПНЧЭ
7	бел	If	'123abc45de'	3	МГ; ВГ	СЭКЗ; СЧЭ
8	бел	switch-case	'123abcde45'	5	МГ; ДМ	СЭКЗ; ПЭНН
9	бел	массив	'12ab34cde5'	7	МГ; ВсГ	СЭКЗ; ПНЧЭ
10	нем	If	'123abc45de'	9	ВсГ; ВГ	ПНЧЭ; СЧЭ
11	нем	switch-case	'1ab23c45de'	4	ВсГ; ДМ	ПННЭ; ПЭНН
12	нем	массив	'1ab2345cde'	7	ВсГ; МГ	ПННЭ; СЭКЗ

Контрольные вопросы

1 Какие варианты использования оператора ***if*** Вы знаете?
2 Какие операторы ветвлений Вам известны?
3 Прокомментируйте назначение и структуру оператора ***switch-case***.
4 Для чего предназначена инструкция ***break***?
5 Какие операторы цикла Вы знаете?
6 Прокомментируйте синтаксис оператора ***for***.
7 Для чего предназначен оператор ***for in***?
8 Как реализовать досрочный выход из цикла?
9 Прокомментируйте синтаксис оператора ***while***.
10 Для чего используется инструкция continue?

5 Лабораторная работа № 5. Изучение методов JavaScript

Методы ***JavaScript*** – это действия, которые могут выполняться над объектами, в то же время это свойства, содержащие определение функции, например,

```
var person = {
   firstName: "John",
   lastName : "Doe",
   id : 5566,
   fullName : function() {
      return this.firstName + " " + this.lastName;
   }
};
```

Ключевое слово ***this*** в данном примере относится к владельцу функции, т. е. к объекту ***Person***, который владеет функцией ***function()***.

Методом объекта ***JavaScript*** может быть только функция, а значением свойства объекта – любой тип данных, за исключением функции.

Основные методы ***JavaScript***. Методы строк. Строка в ***JavaScript*** является одновременно и объектом ***string*** и переменной, поэтому может быть создана двумя способами:

```
st1 = new String("Строка – это объект")
st2 = "Строка – это переменная"
```

В JavaScript используются следующие методы строк:

charAt() – извлекает из строки символ, находящийся в указанной позиции;

charCodeAt() – возвращает код юникода символа, находящегося в указанной позиции (16-разрядное целое число между 0 и 65 535);

concat() – выполняет конкатенацию одного или нескольких значений со строкой, преобразует все аргументы в строки и добавляет их по порядку в конец строки;

indexOf (подстрока, начало) – выполняет поиск в строке от начала к концу;

lastIndexOf() – выполняет поиск символа или подстроки в строке с конца;

match() – выполняет поиск по шаблону с помощью регулярного выражения.

Для выделения нескольких символов строки используется метод ***substr()***.

Замена подстроки текста выполняется методом ***replace()***, построенным на использовании регулярных выражений.

Методы вывода:

alert() – выводит модальное окно с сообщением;

confirm() – выводит сообщение в окне с двумя кнопками: ***ОК*** и ***ОТМЕНА*** и возвращает выбор посетителя;

prompt() – выводит окно с указанным текстом и полем для пользовательского ввода;

setInterval() – выполняет код или функцию через указанный интервал времени.

Метод ***document.write()*** выводит на страницу переданные ему аргументы.

Глобальные методы ***JavaScript***:

alert() – выводит модальное окно с сообщением;

clearInterval() – останавливает выполнение кода, заданное ***setInterval***;

clearTimeout() – отменяет выполнение кода, заданное ***setTimeout()***;

confirm() – выводит сообщение в окне с двумя кнопками: ***ОК*** и ***ОТМЕНА*** и возвращает выбор посетителя;

decodeURI() – раскодирует ***URI***, закодированный при помощи ***encodeURI***;

decodeURIComponent() – раскодирует ***URI***, закодированный при помощи encodeURIComponent();

encodeURI() – кодирует ***URI***, заменяя каждое вхождение определенных символов на escape-последовательности, представляющие символы в кодировке UTF-8;

encodeURIComponent() – кодирует компоненту ***URI***, заменяя определенные символы на соответствующие ***UTF***-8 escape-последовательности;

eval() – выполняет строку ***JavaScript***-кода без привязки к конкретному объекту;

isFinite() – проверяет, является ли аргумент конечным числом;

isNaN() – проверяет, является ли аргумент ***NaN***.

Порядок выполнения работы.

Выполнить задания.

1 Заполнить массив из 14 элементов вещественными числами, используя заданную в колонке 2 таблицы 5.1 функцию. Рассортировать массив в указанном в колонке 3 порядке.

2 Вывести текущую дату в заданном в колонке 4 формате и определить:

– используя функцию ***DataParse,*** количество миллисекунд, прошедших с 01.01.1970 г. по текущий момент;

– используя метод ***getTime,*** количество секунд от 01.01.1970 г. по текущий момент;

– исплользуя метод ***getDay***, номер и название дня Вашего рождения.

3 Вычислить значение элемента, заданного в колонке 5.

Таблица 5.1 – Варианты заданий

Номер варианта	Задание 1		Задание 2	Задание 3
	Функция	Сортировать	Дата	Вычислить
1	2	3	4	5
1	sin(x)	По возрастанию	Год, месяц, день	Площадь усеченного конуса
2	cos(x)	По убыванию	Месяц, день, час	Площадь трапеции
3	tg(x)	По возрастанию	Час, минута, секунда	Площадь параллелограмма
4	ctg(x)	По убыванию	День, час, минута	Макс. значение среди трех
5	sin(x/2)	По возрастанию	Год, месяц, день	Макс. значение среди пяти
6	cos(x/2)	По убыванию	Месяц, день, час	Объем конуса
7	tg(x/2)	По возрастанию	Час, минута, секунда	Объем усеченного конуса
8	ctg(x/2)	По убыванию	День, час, минута	Объем пирамиды
9	sin(x^2)	По возрастанию	Год, месяц, день	Объем усеченной пирамиды
10	cos(x^2)	По убыванию	Месяц, день, час	Площадь шестиугольника
11	tg(x^2)	По возрастанию	Час, минута, секунда	Площадь кольца
12	ctg(x^2)	По убыванию	День, час, минута	Площадь цилиндра

Контрольные вопросы

1 Какие методы вывода в ***JavaScript*** Вы знаете?
2 Для чего используются методы ***alert()***, ***prompt()*** и ***document.write()***?
3 Какие методы объекта ***Math*** Вы знаете?
4 Прокомментируйте назначение метода ***document.getElementById(id)***.
5 Какие два способа создания строки Вы знаете?
6 Что понимается под методом в ***JavaScript***?
7 Прокомментируйте назначение функции ***Date.parse***.
8 Какой метод используется для поиска номера символа строки?
9 Через какой объект выполняется работа с датами в ***JavaScript***?
10 Какой метод используется для округления чисел?

6 Лабораторная работа № 6. Изучение работы с массивами на JavaScript

Массивы в ***JavaScript*** являются нетипизированными, т. е. элементы одного и того же массива могут иметь разные типы. Элементы массива могут быть объектами или другими массивами, что позволяет создавать сложные структуры данных, такие как массивы объектов и массивы массивов.

Отсчет индексов массивов в языке ***JavaScript*** начинается с нуля, и для них используются 32-битные целые числа. Массивы в ***JavaScript*** являются динамическими: они могут увеличиваться и уменьшаться в размерах по мере необходимости, поэтому нет необходимости объявлять фиксированные размеры

массивов при их создании или повторно распределять память при изменении их размеров.

Создать массив проще всего с помощью литерала, который представляет собой простой список разделенных запятыми элементов в квадратных скобках. Значениями литерала массива могут быть константы, выражения, литералы объектов:

```
var empty = [];                 // Пустой массив
var numbers = [2, 3, 5, 7, 11];         // Массив с пятью числовыми элементами
var misc = [ 1.1, true, "a", ];  // 3 элемента разных типов + завершающая запятая
var base = 1024;
var table = [base, base+1, base+2, base+3];  // Массив с переменными
var arrObj = [[1,{x:1, y:2}], [2, {x:3, y:4}]];  // 2 массива внутри, содержащие объекты
```

Синтаксис литералов массивов позволяет вставлять необязательную завершающую запятую, т. е. литерал [,,] соответствует массиву с двумя элементами, а не с тремя.

Другой способ создания массива – конструктор ***Array()***, который можно вызвать тремя разными способами: без аргументов; с единственным числовым аргументом, определяющим длину массива; с явным указанием значений первых двух или более элементов или одного нечислового элемента:

```
var arr = new Array();                 // пустой массив, эквивалентный литералу []
var arr = new Array(10);               // пустой массив указанной длины
var arr = new Array(5, 4, 3, 2, 1, "тест");  //массив с явным указанием элементов
```

Массив является специализированной разновидностью объекта, поэтому квадратные скобки, используемые для доступа к его элементам, действуют аналогично доступу к свойствам объекта. Интерпретатор ***JavaScript*** преобразует указанные в скобках числовые индексы в строки, например, индекс 1 – в строку «1», а затем использует эти строки как имена свойств.

Следует четко отличать индексы в массиве от имен свойств объектов. Все индексы массива являются именами свойств, но только свойства с именами, представленными целыми числами, являются индексами. Массивы являются объектами, и к ним можно добавлять свойства с любыми именами. Однако если затрагиваются свойства, которые являются индексами массива, то массивы реагируют на это, обновляя при необходимости значение свойства ***length***.

В качестве индексов массивов допускается использовать отрицательные и нецелые числа. В этом случае числа преобразуются в строки, которые используются как имена свойств.

Добавление и удаление элементов массива. Самый простой способ добавления элементов массива – это присваивание значений его новым индексам. Для добавления одного или более элементов в конец массива можно также использовать метод ***push()***:

```
var arr = [];              // Создать пустой массив
arr.push('zero');          // Добавить значение в конец массива
```

```
arr.push('one',2);      // Добавить еще два значения в массив
```

Добавить элемент в конец массива можно также, присвоив значение элементу ***arr[arr.length]***. Для вставки элемента в начало массива можно использовать метод ***unshift()***, при этом существующие элементы в массиве смещаются в позиции с более высокими индексами.

Удалять элементы массива можно как обычные свойства объекта – с помощью оператора ***delete***:

```
var arr = [1,2,'three'];
delete arr[2];
2 in arr;               // false, индекс 2 в массиве не определен
arr.length;             // 3: оператор delete не изменяет свойство length массива
```

Многомерные массивы. ***JavaScript*** не поддерживает настоящие многомерные массивы, но позволяет имитировать их при помощи массива из массивов. Для доступа к элементу данных в массиве массивов достаточно дважды использовать оператор [ ].

Методы класса Array.

Array.join() – преобразует все элементы массива в строки, объединяет их и возвращает получившуюся строку.

Array.reverse() – меняет порядок следования элементов в массиве на обратный и возвращает переупорядоченный массив.

Array.sort() – сортирует элементы в исходном массиве и возвращает отсортированный массив.

Array.concat() – создает и возвращает новый массив, содержащий элементы исходного массива, для которого был вызван метод concat(), и значения всех переданных ему аргументов.

Array.slice() – возвращает фрагмент, или подмассив указанного массива.

Порядок выполнения работы.

Выполнить задания, варианты которых приведены в таблице 6.1.

1 Создать одномерный массив ***R*** из ***k*** элементов, указанных в колонке 2. Добавить ***m*** числовых элементов в его начало и ***n*** текстовых – в конец.

2 Создать двухмерный массив размерностью ***m*** × ***n*** (колонка 4) и заполнить случайными равномерно распределенными числами. Строки с четными номерами рассортировать.

3 Создать двухмерный массив размерностью ***k*** × ***m***, содержащий текстовые и числовые элементы. Используя метод ***Array.join()***, преобразовать все числовые элементы массива в строки; изменить порядок следования элементов массива на обратный.

4 Создать двухмерный текстовый массив размерностью ***m*** × ***k***. Используя методы ***unshift()*** и ***push()***, добавить ***k*** – 4 строки в начало массива и с помощью методов ***shift()*** и ***pop()*** удалить ***m*** – 3 строк из конца заданного массива.

Таблица 6.1 – Варианты заданий

Номер варианта	Задание 1		Задание 2	Задание 3		Задание 4
	строка	*k*, *m*, *n*	*m*, *n*	сортировать	*k*, *m*	*m*, *k*
1	2	3	4	5	6	7
1	['*a*', '*b*', '*c*']	3**,** 2, 4	4, 6	По возрастанию	7, 8	6, 9
2	[1, 2, 3]	4, 3, 2	5, 8	По убыванию	6, 5	7, 6
3	['*p*', '*n*', '*k*']	2**,** 1, 5	3, 5	По возрастанию	8, 6	8, 9
4	[7, 5, 8]	4**,** 2, 3	4, 6	По убыванию	5, 8	5,7
5	['*r*', '*s*', '*t*']	3**,** 4, 4	5, 7	По возрастанию	6, 7	5, 8
6	[2, 6, 8]	5**,** 3, 2	4, 6	По убыванию	5, 8	5, 9
7	['*k*', '*m*', '*s*']	4**,** 1, 3	5, 8	По возрастанию	8, 7	6, 5
8	[3, 7, 5]	2**,** 5,4	3, 6	По убыванию	6, 8	6, 7
9	['*q*', '*z*', '*s*']	5**,** 1, 3	4, 5	По возрастанию	5, 6	6, 8
10	[2, 9, 4]	3**,** 4, 5	5, 7	По убыванию	8, 8	6, 6
11	['*t*', '*r*', '*n*']	4**,** 2, 3	2, 6	По возрастанию	7, 7	7, 6
12	[4, 7, 2]	3**,** 5,2	3, 8	По убыванию	6, 5	7, 7

Контрольные вопросы

1 Что такое динамический массив в ***JavaScript***?
2 Какие методы создания массивов Вы знаете?
3 Что может быть элементом массива в ***JavaScript***?
4 С какого значения ведется отсчет индексов массивов в ***JavaScript***?
5 Как построить доступ к элементу массива?
6 Какой тип может иметь индекс элемента массива?
7 Как понимать: «массив – специализированная разновидность объекта»?
8 Как в ***JavaScript*** поддерживается работа с многомерными массивами?
9 Какие методы класса ***Array*** Вы знаете?
10 Какие методы добавления и удаления элементов массива Вы знаете?

7 Лабораторная работа № 7. Изучение работы с элементами управления JavaScript

Язык ***JavaScript*** позволяет создавать сложные web-элементы управления сайтами, среди которых сложные меню, специализированные деревья и сложные сетки, а также два специальных – генератор всплывающих окон и динамически меняющаяся кнопка.

Всплывающие (***popup-***) окна – это один из способов, позволяющих показать пользователю дополнительный контент.

В недавнем прошлом всплывающими окнами злоупотребляли многие сайты, нацеленные на показ рекламы, и загружали пользователей множеством объявлений. Поэтому современные браузеры блокируют всплывающие окна.

Всплывающее окно достаточно просто отображается с помощью функции ***window.open()*** в блоке ***JavaScript***:

```
window.open('http://www.google.com', 'myWindow',
   'toolbar=0, height=500, width=800, resizable=1, scrollbars=1');
window.focus();
```

Функция ***window.open()*** принимает параметры: ссылка на новую страницу и имя фрейма окна, в которое позже должен быть загружен новый документ посредством другой ссылки. Третий параметр – разделенная запятыми строка, конфигурирующая стиль и размер всплывающего окна с помощью атрибутов:

– ***height*** – высота и ***width*** – ширина в пикселях;

– ***toolbar*** – панель инструментов и ***menuBar*** – строка меню, которые могут быть установлены в 1 или 0, в зависимости от того, требуется ли отображение этих элементов;

– ***resizable*** = 1 – рамка изменяемого размера, = 0 – фиксированного;

– ***scrollbars*** = 1, если требуются линейки прокрутки, = 0 – если нет.

Чтобы закрыть ***popup***-окно, необходимо вызвать функцию ***newWindow.close()***. Метод ***close()*** можно вызвать для любого объекта ***window***, но ***window.close()*** игнорируется почти всеми браузерами, если окно было открыто не с помощью ***window.open()***.

Эффективным элементом управления в ***JavaScript*** является динамически меняющаяся кнопка, которая выводит на экран одно изображение, если она появляется на web-странице впервые, при задержке над ней курсора мыши – другое, при щелчке на этой кнопке – третье.

Для обеспечения такого эффекта кнопка обычно состоит из дескриптора ***<img>***, который обрабатывает ***JavaScript***-события ***onclick***, ***onmouseover*** и ***onmouseout***. Эти события вызывают функции, меняющие изображения для текущей кнопки:

```
function swapImg(id, url) {
   var elm = document.getElementById(id);
   elm.src = url;
}
```

В этом случае сконфигурированный дескриптор ***<img>*** выглядел бы следующим образом:

```
<img id="img" src="activebtn.png" style="cursor:pointer; margin:120px"
   onmouseover="swapImg('img', 'hoverbtn.png')"
   onmouseout="swapImg('img', 'activebtn.png')" />
```

Порядок выполнения работы.

Выполнить задания, варианты которых приведены в таблице 7.1.

1 Написать сценарий, позволяющий продемонстрировать изменения размеров и положения горизонтальной линии на странице ***html***.

2 Написать сценарий формирования анкеты данных сотрудника, указанных в колонке 3.

3 Написать сценарий обработки анкеты слушателя курсов повышения квалификации, содержащей: курс (первый, второй, третий); язык общения с преподавателем; изучаемые дисциплины; продолжительность курса; форму образования (очная, заочная, дистанционная); изучаемые дисциплины; стоимость. В зависимости от этих параметров определяется стоимость отдельного курса и стоимость всего обучения. Выбор значений выполнить с помощью флажков и выпадающего меню.

Таблица 7.1 – Варианты заданий

Номер вари-анта	Задание 1: длина линии, %; толщина, пикс	Задание 2 (данные сотрудника): МР – место рождения; НАЦ – национальность; ОБР – образование; ДР – дата рождения; СП – семейное положение	Задание 3 (поля выбора): ПР – продолжительность; ФО – форма образования; ФИОП – ФИО преподавателя; ИД – изучаемые дисциплины
1	2	3	4
1	45 %; 2 пикс	Пол, МР, ОБР	Курс, язык, ПР, стоимость
2	72 %, 3 пикс	ДР, пол, должность.	Курс, язык, ФО
3	85 %, 4 пикс	МР, возраст, пол	Курс, язык, ФИОП, стоимость
4	57 %, 2 пикс	МР, должность, пол	Курс, язык, ПР, ИД
5	80 %, 3 пикс	Пол, должность, ДР	Курс, язык, ФО, стоимость
6	90 %, 2 пикс	НАЦ, пол, возраст	Курс, язык, ФИОП, ИД
7	50 %, 2 пикс	СП, пол, должность	Курс, язык, ФО, стоимость
8	70 %, 3 пикс	Пол, НАЦ, возраст	Курс, язык, ПР, ИД
9	80 %, 4 пикс	МР, возраст, должность	Курс, язык, ФИОП, стоимость
10	35 %, 2 пикс	ДР. должность, возраст	Курс, язык, ПР, ИД
11	45 %, 3 пикс	НАЦ, должность, образование	Курс, язык, ФО, стоимость
12	75 %, 4 пикс	МР, возраст, должность	Курс, язык, ФИОП, ИД

Контрольные вопросы

1 Для чего используется генератор всплывающих (***popup***-) окон?
2 Что такое динамически меняющаяся кнопка?
3 Какие web-элементы управления сайтами Вы знаете?
4 Почему современные браузеры блокируют всплывающие окна?
5 Для чего используется функция ***window.open()***?
6 Как построить динамически изменяющуюся кнопку?
7 Какие параметры содержит функция ***window.open()***?
8 Как закрыть ***popup***-окно?
9 Где используется и для чего свойство ***z-index***?
10 Как работают динамически изменяющиеся кнопки?

8 Лабораторная работа № 8. Изучение работы с изображениями на языке JavaScript

Изображения в ***html***-документах представляются в виде массива, что позволяет адресоваться к ним. Они имеют определенные свойства и рассматриваются как объекты ***image***, к которым можно обращаться из языка ***JavaScript***. Например, можно определить размер изображения, обратившись к его свойствам ***width*** и ***height***. То есть по записи ***document.images[0].width*** можно определить ширину первого изображения в пикселях на web-странице.

Объект ***Image*** позволяет вносить изменения в графические образы на web-странице, что позволяет создавать мультипликацию. Для отслеживания индекса всех изображений web-страницы следует назначить им имена с помощью тега

<img src="img.gif" name="myImage" width=100 height=100>

Тогда для обращения к изображению необходимо написать ***document.myImage*** или ***document.images["myImage"]***.

Смена изображения на web-странице выполняется с использованием атрибута ***src***, который содержит адрес представленного изображения и позволяет назначить ему новый адрес. В результате изображение будет загружено с этого нового адреса, заменяя на web-странице старое изображение:

<img src="img1.gif" name="myImage" width=100 height=100>

Здесь загружается изображение ***img1.gif*** и получает имя ***myImage***. В следующей строке изображение ***img1.gif*** заменяется на новое – ***img2.gif***:

document.myImage.src= "img2.src".

При этом новое изображение всегда получает тот же размер, который был у старого, и уже невозможно будет изменить размер поля, в котором это изображение размещается.

Одной их замечательных возможностей браузеров являются слои, позволяющие позиционировать изображения, организовывать их перемещение и делать их невидимыми.

Для создания слоев можно использовать тег <***layer***> или тег <***ilayer***>. При этом можно воспользоваться следующими параметрами:

name = "***layerName***" – название слоя;
left = ***xPosition*** – абсцисса левого верхнего угла;
top = ***yPosition*** – ордината левого верхнего угла;
z-index = ***layerIndex*** – номер индекса для слоя;
width = ***layerWidth*** – ширина слоя в пикселах;
clip = "***x1_offset, y1_offset, x2_offset, y2_offset***" – видимая область слоя;
above = "***layerName***" – определяет, какой слой окажется под текущим;

below = "***layerName***" – определяет, какой слой окажется над текущим;
visibility = ***show|hide|inherit*** – видимость этого слоя;
bgcolor = "***rgbColor***" – цвет фона, или название стандартного цвета, или ***rgb***-запись;
background = "***imageURL***" – фоновая картинка.

Тег <***layer***> используется для слоев, которые можно точно позиционировать. Если не указать положение слоя с помощью параметров ***left*** и ***top***, то по умолчанию он помещается в верхний левый угол окна.

Тег <***ilayer***> создает слой, положение которого определяется при формировании документа.

Поверх изображения или под ним можно показать текст.

Свойства слоев можно изменять с помощью скриптов на ***JavaScript***.

Следующий пример демонстрирует как скрипт может реагировать на сигналы о нажатии клавиш.

```
<html>
    <script language="JavaScript">
      window.captureEvents(Event.KEYPRESS);
      window.onkeypress= pressed;
      function pressed(e) {
        alert("Key pressed! ASCII-value: " + e.which);  }
    </script>
</html>
```

Порядок выполнения работы.

Создать ***html***-документ, элементы и параметры которого указаны в таблице 8.1.

Таблица 8.1 – Варианты заданий

Номер вари-анта	Количество графических элементов	Ориентация группы элементов	Изображения меняются местами	При наведении курсора
1	3	Горизонтально	При наведении курсора	Увеличить на 25 %
2	4	Горизонтально	При наведении курсора	Увеличить на 40 %
3	3	Вертикально	По щелчку мыши	Уменьшить на 30 %
4	4	Вертикально	По щелчку мыши	Уменьшить на 40 %
5	3	Горизонтально	При наведении курсора	Увеличить на 35 %
6	4	Горизонтально	При наведении курсора	Увеличить на 45 %
7	3	Вертикально	По щелчку мыши	Уменьшить на 35 %
8	4	Вертикально	По щелчку мыши	Уменьшить на 50 %
9	3	Горизонтально	При наведении курсора	Увеличить на 30 %
10	4	Горизонтально	При наведении курсора	Увеличить на 45 %
11	3	Вертикально	По щелчку мыши	Уменьшить на 20 %
12	4	Вертикально	По щелчку мыши	Уменьшить на 45 %

Контрольные вопросы

1 Как в ***html***-документах представляются изображения?
2 Как из ***JavaScript*** можно адресоваться к изображениям?
3 Для чего предназначен атрибут ***src*** тега <***img***>?
4 Для чего предназначен объект ***Image***?
5 Какие условия следует соблюдать, чтобы скрипт смены изображений сохранял свою гибкость?
6 Какие свойства определяют размеры объекта ***image***?
7 Какой объект позволяет вносить изменения в графические образы на web-странице для создания мультипликации?
8 Как изменить свойство слоя?
9 Что позволяет делать изображения в браузерах невидимыми?
10 Какой тег используется для создания слоев?

9 Лабораторная работа № 9. Изучение технологии обработки событий на языке JavaScript

Событие – это сигнал от браузера о том, что что-то произошло, например, ***load*** – загрузка страницы и ***unload*** – выгрузка ее. Клиентские программы на языке ***JavaScript*** основаны на модели программирования, когда выполнение программы управляется событиями. При таком стиле программирования web-браузер генерирует событие, когда с документом или некоторым его элементом что-то происходит. Например, web-браузер генерирует событие, когда завершает загрузку документа, когда пользователь наводит указатель мыши на гиперссылку или нажимает клавишу на клавиатуре.

События мыши: ***click*** – клик; ***dblclick*** – двойной клик; ***mouseover*** – наведение курсора мыши на элемент; ***mousemove*** – перемещение курсора мыши над элементом; ***mouseout*** – уведение курсора мыши с элемента; ***mousedown*** – нажатие левой кнопки мыши; ***mouseup*** – отпускание левой кнопки мыши; ***contextmenu*** – нажатие правой кнопки мыши и вывод контекстного меню.

Для того чтобы обратить внимание пользователя на определённый элемент ***html***-документа, можно менять свойства этого элемента при попадании на него курсора мышки, а при снятии курсора восстанавливать прежние значения свойств. Например, можно менять цвет или размер элемента. Попадание курсора мышки на элемент фиксируется событием ***onMouseOver***. Парное для него событие ***onMouseOut*** происходит при снятии курсора мышки с элемента.

Другая пара событий ***onMouseDown*** и ***OnMouseUp*** происходит при нажатии и отпускании левой кнопки мышки. Эту пару событий удобно применять для изменения свойств элементов или замены элементов на время удержания кнопки мышки нажатой.

События клавиатуры: ***keyup*** – возникает при отпускании клавиши клавиатуры; ***keydown*** – возникает при нажатии клавиши клавиатуры и длится,

пока нажата клавиша; ***keypress*** – возникает при нажатии клавиши клавиатуры, но после события ***keydown*** и до события ***keyup***.

Для того чтобы написать ответную реакцию на событие, создают обработчик события ***event handler***, который, как правило, представляет собой функцию.

Назначить обработчик события можно в виде атрибута элемента, безымянной функции, именованной функции или с помощью метода ***addEventListener().***

Тип события – это строка, определяющая тип действия, вызвавшего событие. Тип ***mousemove***, например, означает, что пользователь переместил указатель мыши, тип ***keydown*** – что была нажата клавиша на клавиатуре, а тип ***load*** – что завершилась загрузка документа, или какого-либо другого ресурса из сети.

Обработчик события – это функция, которая обрабатывает событие или откликается на него. Приложения должны зарегистрировать свои функции обработчиков событий в web-браузере, указав тип события и цель.

Реакция на событие в отдельном элементе. Так как в объектной модели объекты могут быть вложены друг в друга, то событие, происходящее в дочернем объекте, одновременно происходит и в родительском. ***JavaScript*** предоставляет различные способы локализации влияния события на иерархию объектов. Простейшей способ локализации заключается в размещении сценария в теге, на который должно воздействовать событие.

Порядок выполнения работы.

Создать ***html***-документ, содержащий строку и изображение с подписью, параметры которого указаны в таблице 9.1.

Таблица 9.1 – Варианты заданий

Номер вари-анта	Создать строку, при наведении курсора на которую меняется			Создать изображение с надписью на нем и подписью	
	шрифт	цвет шрифта	цвет фона	при щелчке на изображении меняется	цвет при щелчке на подписи
1	Увеличить до 48 pt	Белый	Голубой	Изображение и надпись	Синий
2	Увеличить до 64 pt	Синий	Белый	Цвет шрифта и надпись	Красный
3	Увеличить до 36 pt	Красный	Голубой	Цвет фона и надпись	Зеленый
4	Увеличить до 42 pt	Зеленый	Белый	Цвет шрифта и изображение	Голубой
5	Увеличить до 52 pt	Синий	Зеленый	Цвет фона и изображение	Желтый
6	Увеличить до 56 pt	Белый	Голубой	Фон и цвет надписи	Зеленый
7	Уменьшить на 12 pt	Синий	Белый	Изображение и надпись	Синий
8	Уменьшить на 16 pt	Красный	Голубой	Цвет шрифта и надпись	Желтый
9	Уменьшить на 24 pt	Зеленый	Белый	Цвет фона и надпись	Красный
10	Уменьшить на 20 pt	Синий	Зеленый	Цвет шрифта и изображение	Голубой
11	Уменьшить на 10 pt	Красный	Голубой	Цвет фона и изображение	Зеленый
12	Уменьшить на 14 pt	Зеленый	Белый	Фон и цвет надписи	Синий

Контрольные вопросы

1 Как назначить обработчик событий?
2 Что означает тип события ***load***?
3 Какие события ***JavaScript*** относятся к системным?
4 Какие виды событий используются в ***JavaScript***?
5 Какие события клавиатуры Вы знаете?
6 Что понимается под типом события?
7 Какие события мыши Вы знаете?
8 Что означают типы событий ***mousemove*** и ***keydown***?
9 Что понимается под целью события?
10 Что такое обработчик событий?

10 Лабораторная работа № 10. Изучение основных методов Jquery

Методы ***Jquery*** позволяют манипулировать содержимым web-страницы. Они присваивают элементам, отобранным в ***Jquery***-объектах, заданные действия. В результате этого происходит динамическое изменение элементов и их содержимого.

Каждый метод ***Jquery*** либо сам что-либо возвращает, либо получает параметр и выполняет указанные в параметре действия.

В общем виде синтаксис для вызова метода ***Jquery*** имеет следующий вид:

$("селектор").имяМетода(параметры);

Динамическое изменение элементов web-страниц. Библиотека ***Jquery*** упрощает процесс отбора элементов ***html***-страниц. С помощью методов ***Jquery*** производятся манипуляции с объектной моделью документа DOM. Чтобы отобрать группу элементов, нужно передать селектор функции ***Jquery***. В качестве селектора элемента может выступать сам элемент, его идентификатор или класс, а также комбинация селекторов: ***$("a")***; ***$("#some-id")***; ***$(".someclass")***; ***$("header > ul:has(a)")***.

Функция ***$()*** возвращает объект ***Jquery***, содержащий массив элементов DOM – так называемый обернутый набор, соответствующий указанному селектору. Большинство методов по завершении действий возвращает первоначальный набор элементов.

Jquery – это одна из наиболее известных библиотек, написанная на языке ***JavaScript*** для упрощения программирования web-страниц. Это файл с расширением ***.js***, который подключается к web-странице как фрагмент скрипта и загружается в браузер вместе с web-страницей.

Чтобы включить ***Jquery*** в web-страницу, достаточно скачать последнюю версию библиотеки, например, файл ***jquery-1.9.1.js*** с сайта ***jquery.com***, положить

его в ту же папку, где лежит текст web-страницы, а в текст web-страницы вставить <***script src="jquery-1.9.1.js"></script***>

Файл ***jquery-1.9.1.js*** при этом имеет объем более 200 Кбайт, что может замедлять загрузку web-страницы в браузере пользователя.

Если web-страница маленькая и важно, чтобы все «летало» и загрузка библиотеки ***Jquery*** ничего не замедляла, то существует альтернативный метод загрузки ***Jquery*** – с сайта ***Google***:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js">
</script>
```

Второе важное характерное для ***Jquery*** применение состоит в создании ***AJAX***-элементов, т. е. тех элементов страницы, которые отсылают на сервер данные и получают ответ без перезагрузки страницы. К таким элементам можно отнести форму «Управление корзиной для интернет магазина», пагинацию (нумерацию страниц сайта), вывод информера погоды и многое другое.

Порядок выполнения работы.

Используя библиотеку ***Jquery*** создать сайт. Варианты заданий указаны в таблице 10.1.

Таблица 10.1 – Варианты заданий

Номер варианта	Используя библиотеку ***Jquery*** создать сайт, содержащий
1	Выпадающее меню
2	Плавающее меню
3	Фотогалерею
4	Всплывающие окна
5	Слайдеры (блоками на странице, в пределах которых с установленной периодичностью происходит демонстрация анонсов новостей, статей или изображений)
6	Перемещающиеся блоки
7	Изменение прозрачности элементов
8	Подсвечивание текста
9	Переливание цвета текста разными оттенками
10	Плавающее меню
11	Всплывающие окна
12	Изменение прозрачности элементов

Контрольные вопросы

1 Какие действия выполняют методы ***Jquery***?
2 Прокомментируйте синтаксис вызова методов ***jquery***.
3 Что понимается под селектором ***jquery***?
4 Что такое ***Jquery***?
5 Какие основные правила использования ***Jquery*** Вы знаете?

6 Какие основные методы ***Jquery*** Вы знаете?
7 Как подключить библиотеку ***Jquery*** к web-странице?
8 Где найти файл библиотеки ***Jquery***?
9 Что такое слайдер?
10 Что такое альтернативный метод загрузки ***Jquery***?

11 Лабораторная работа № 11. Изучение основных событий Jquery

События ***Jquery***, представляющие собой момент, в который что-либо происходит, например щелчок кнопки мыши, помогают сделать web-страницы интерактивными, реагирующими на простейшие действия пользователя.

Момент, в который произошло событие, называется запуском события. События могут срабатывать при выполнении различных операций с web-страницей. Помимо этого и сам браузер может стать источником событий.

Управление web-страницей производится с помощью следующих событий мыши; документа/окна; форм; клавиатуры; ***Jquery***.

События мыши:

.click() – запускается при нажатии и отпускании кнопки мыши, применяется к ссылкам, картинкам, кнопкам, абзацам, блокам и т. д.;

.dblclick() – запускается при двойном нажатии и отпускании кнопки мыши, например, при открытии какой-либо папки;

.mousedown() – происходит во время нажатия кнопки мыши, например, при перетаскивании элементов;

.mousemove() – запускается при перемещении указателя мыши по элементу;

.mouseout() – запускается при отпускании кнопки мыши.

События документа/окна:

.load() – запускается, когда браузер загрузит все файлы web-страницы: ***html***-файлы, внешние CSS- и ***JavaScript***-файлы, медиафайлы;

.resize() – запускается, когда пользователь изменяет размер окна браузера;

.scroll() – запускается, когда пользователь использует полосы прокрутки, либо прокручивает web-страницу с помощью колесика мыши, либо использует для этих целей клавиши клавиатуры (***pgup***, ***pgdn***, ***home***, ***end***);

.unload() – запускается, когда пользователь собирается покинуть страницу, щелкая по ссылке для перехода на другую страницу, закрывает вкладку страницы или окно браузера.

События форм:

.blur() – запускается, когда поле формы выводится из фокуса, например, при переходе в другое поле формы;

.change() – запускается при изменении статуса поля формы, например при выборе пункта из выпадающего меню;

.focus() – запускается при переходе в поле формы, при щелчке на нем кнопкой мыши или клавишей табуляции;

.reset() – позволяет вернуть форму в первоначальное состояние, отменив сделанные изменения;

.select() – запускается при выделении текста внутри текстового поля формы;

.submit() – запускается при отправлении заполненной формы с помощью щелчка по кнопке «Отправить» или нажатии клавиши ***Enter***, когда курсор помещен в текстовом поле.

События клавиатуры:

.keydown() – запускается при нажатии клавиши перед событием ***keypress***;

.keypress() – запускается при нажатии на клавишу до тех пор, пока клавиша не будет отпущена;

.keyup() – запускается при отпускании клавиши.

События ***jQuery***:

.hover() – позволяет одновременно решить две задачи, связанные с событием наведения указателя мыши и событием снятия указателя мыши в отношении выбранного объекта;

.toggle() – работает аналогично событию ***hover()***, с разницей в том, что оно запускается от щелчка кнопкой мыши. Например, можно открыть выпадающее меню одним щелчком и скрыть вторым.

Объект события: при запуске события браузер сохраняет информацию о нём в объекте события, который содержит данные, собранные в момент, когда событие произошло. Обработка события происходит с помощью функции, при этом объект передается функции как аргумент–переменная ***evt***.

Объект события имеет различные свойства, наиболее распространённые из которых следующие:

pageX – расстояние (***px***) от указателя мыши до левого края окна браузера;

pageY – расстояние (***px***) от указателя мыши до верхнего края окна браузера;

screen – расстояние (***px***) от указателя мыши до левого края монитора;

screenY – расстояние (***px***) от указателя мыши до верхнего края монитора;

shiftKey – ***true***, если была нажата клавиша ***shift***, когда происходило событие;

which – используется для определения числового кода нажатой клавиши (вместе с shiftKey);

target – означает, что по объекту события щелкнули кнопкой мыши (например, для события ***click())***;

data – объект, использованный с функцией ***bind()*** для передачи данных функции, управляющей событием.

Порядок выполнения работы.

Используя события, указанные в таблице 11.1, и библиотеку ***Jquery***, создать сайт.

Таблица 11.1 – Варианты заданий

Номер варианта	Используя библиотеку ***Jquery*****,** создать сайт, содержащий
1	События мыши .click() и .mousedown()
2	События мыши .dblclick() и .mousemove()
3	События документа/окна .load() и .scroll()
4	События документа/окна .resize() и .unload()
5	События форм. blur(), .focus() и .select()
6	События форм .change(), .reset() и .submit()
7	События клавиатуры .keydown() и .keyup()
8	События клавиатуры .keypress() и .keyup()
9	Событие jQuery .hover()
10	Событие jQuery .toggle()
11	События мыши .click() и .mousemove()
12	События форм. blur(), .focus() и .submit()

Контрольные вопросы

1 Какие события мыши Вы знаете?
2 Что такое объект события?
3 Какие события документа/окна Вы знаете?
4 Что представляют собой события ***Jquery***?
5 Какие события клавиатуры Вы знаете?
6 Какие события форм Вы знаете?
7 Какие события ***Jquery*** Вы знаете?
8 В каком объекте браузер сохраняет информацию о событии при его запуске?
9 Какие свойства объекта события Вы знаете?
10 С помощью каких событий производится управление web-страницей?

12 Лабораторная работа № 12. Установка локального сервера

При разработке и отладке серверных приложений используются локальные сервера, наиболее популярным среди которых является web-сервер ***Apache***.

Связь внешней программы с web-сервером выполняется с использованием CGI (***Common Gateway Interface*** – общий интерфейс шлюза), а программу, позволяющую использовать консоль ввода и вывода для взаимодействия с клиентом и работающую по интерфейсу CGI, принято называть шлюзом, но используется также и название «скрипт» (сценарий) или «CGI-программа».

Для разработки серверных приложений на языке PHP используется интерпретатор PHP, который представляет собой внешнюю CGI-программу, либо динамическую библиотеку, которую необходимо подключить к web-серверу, чтобы вместо кода PHP-скриптов клиенту выдавались результаты ее выполнения.

Традиционно совместно с web-сервером ***Apache*** и интерпретатором PHP используется СУБД ***MySQL***.

Так как и язык программирования PHP, и web-сервер ***Apache***, и ***MySQL***-сервер первоначально разработаны для UNIX-подобных операционных систем, их настройка и администрирование сводятся к редактированию конфигурационных файлов. Такой подход часто сбивает с толку программистов, не имеющих опыта работы в UNIX. Чтобы не завязнуть в многочисленных настройках серверов, правилах их связывания друг с другом, web-разработчики прибегают к готовым пакетам, где вся настройка выполнена профессиональными администраторами. Остается только загрузить и установить готовый пакет, представляющий собой CMS (***Content Management System*** – система управления содержимым), после чего можно сразу приступить к работе с языком программирования. Популярными пакетами, объединяющими PHP, web-сервер Apache и СУБД ***MySQL***, являются ***WordPress***, ***Drupal***, ***Joomla***, ***DataLife Engine***. ***Wamp***, ***Mamp*** и др.

Самой популярной CMS, распространяемой по открытому лицензионному соглашению, является ***WordPress***. По данным web ***Technology Surveys***, на этом движке по состоянию на ноябрь 2018 г. разработано 32,3 % от общего числа существующих сайтов, а также 59,5 % сайтов, использующих CMS. С помощью ***WordPress*** можно создать интернет-магазин, личный блог, корпоративный сайт, информационный портал, отраслевой ресурс, галерею мультимедиа и др.

Принципы построения сайтов в среде ***WordPress*** понятны на интуитивном уровне. После создания и настройки сайта необходимо опубликовать контент, а чтобы сайт был эффективным, контент должен быть качественным и полезным для аудитории, поэтому его необходимо регулярно обновлять, что является самой сложной и ответственной работой.

Одним из наиболее простых в использовании и легко настраиваемых CMS является также ***MAMP***, который доступен по адресу ***https://www.mamp.info/en/downloads/***.

После скачивания ***MAMP*** необходимо кликнуть на иконку установочного пакета, чтобы запустить процесс распаковки и установки ***MAMP*** на компьютер. После всех успешных действий установки появится диалоговое окно локального сервера (рисунок 12.1).

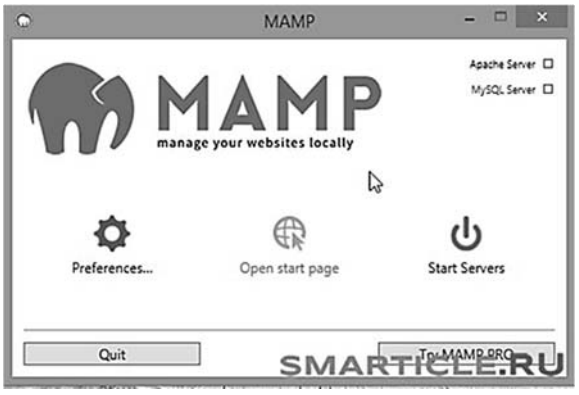


Рисунок 12.1 – Диалоговое окно MAMP

Особого внимания здесь заслуживает ссылка с шестеренкой и надписью ***Preferences*** – Настройки и привилегии, активизация которой выводит окно с пятью вкладками (рисунок 12.2).

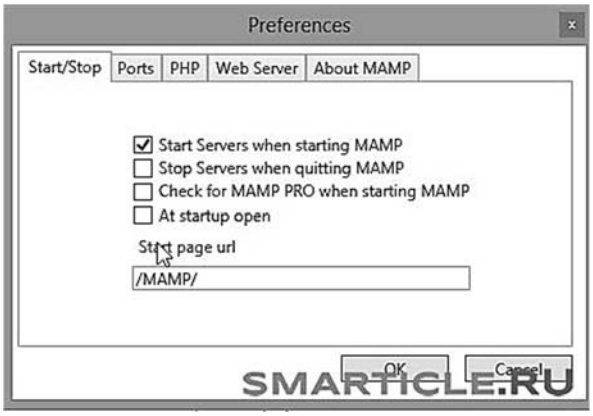


Рисунок 12.2 – Меню *Preferences*

После успешных настроек должны загореться два пункта зеленым цветом – ***Apache*** и ***MySql*** (см. рисунок 12.1), подтверждающих, что сервер работает.

Далее следует перейти на стартовую страницу, нажав на ссылку ***Open Start page***, после чего должен открыться браузер, в адресной строке которого появится локальный путь ***localhost/MAMP***, по которому будет выполняться обращение к файлам сайта.

После этого следует перейти в навигационное меню, в котором необходимо отметить только один раздел ***Tools*** (Инструментарий), где расположена ссылка для доступа в ***phpMyAdmin***.

Вторая важная вкладка необходима для разрешения конфликта между Скайпом – ***Ports*** (Порты): порт ***Апач*** – 80, ***MySql***-порт – 3306.

Остальные вкладки можно не редактировать.

Далее, после проведенных настроек, можно запустить сервер, нажав на ссылку ***Start Servers*** или на ссылку ***Open Start page***, перейти на стартовую страницу и открыть браузер (рисунок 12.3).

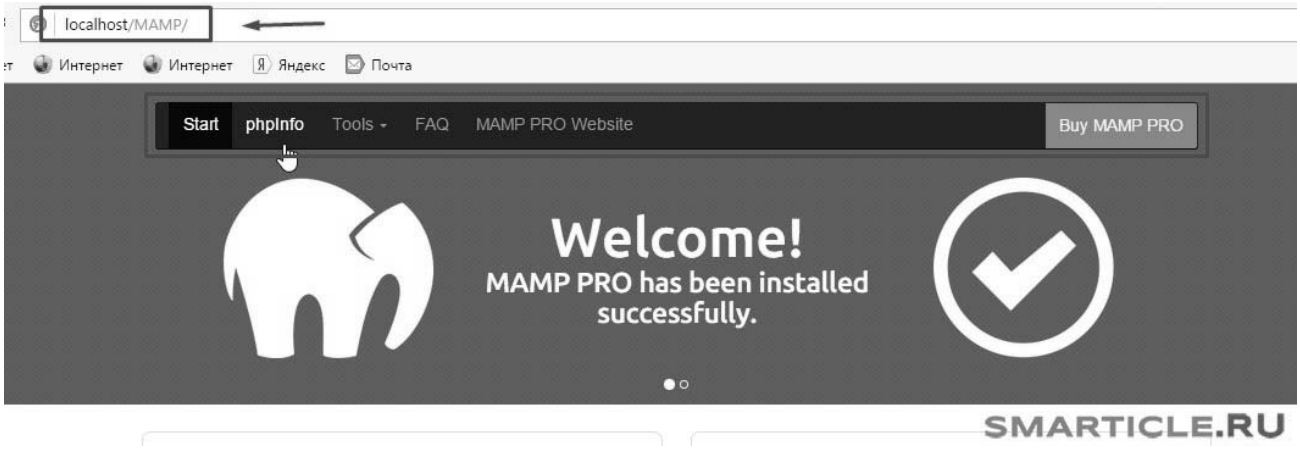


Рисунок 12.3 – Окно браузера

Здесь в адресной строке прописан локальный путь, по которому будет выполняться обращение к файлам сайта – ***localhost/MAMP***.

Далее следует навигационное меню, в котором интересен только один раздел ***Tools*** (Инструментарий). Именно здесь расположена ссылка для доступа в ***phpMyAdmin***.

Порядок выполнения работы.

Выполнить установку и настройку web-сервера ***Apache***, интерпретатора PHP и СУБД MySQL на свой компьютер. Для этого ознакомиться с установкой и настройкой CMS ***MAMP***,

Контрольные вопросы

1 Что такое CMS?
2 Как выбрать хостинг и зарегистрировать домен?
3 Как установить CMS ***WordPress***?
4 Что такое локальный web-сервер?
5 Как создать статическую страницу в ***WordPress***?
6 Для чего используются пакеты ***WAMP*** и ***MAMP***?
7 Как настроить CMS ***WordPress*** и тему?
8 Для чего используется локальный web-сервер ***Apache***?
9 Какие пакеты, объединяющие PHP, web-сервер ***Apache*** и СУБД ***MySQL*** Вы знаете?
10 Можно ли создать сайт без знания ***html*** и языка PHP?

13 Лабораторная работа № 13. Изучение строковых функций языка PHP

В языке PHP используются три способа задания строк: с помощью одинарных кавычек, двойных кавычек и с использованием ***heredoc***-синтаксиса.

Строки, содержащие заключенные в одинарные кавычки переменные и управляющие последовательности специальных символов, не обрабатываются.

Важнейшим свойством строк в двойных кавычках является обработка содержащихся в них переменных.

Определение строк с использованием ***heredoc***-синтаксиса начинается с символа <<< , после которого следует идентификатор. Заканчивается строка этим же идентификатором, который должен начинаться с первой позиции новой строки.

Heredoc-текст ведет себя так же, как и строка в двойных кавычках. Это означает, что в ***heredoc*** нет необходимости экранировать кавычки, но можно использовать управляющие последовательности. Переменные внутри ***heredoc*** также обрабатываются.

Для работы со строками в PHP имеется более ста функций, некоторые из которых приведены в таблица 13.1.

Таблица 13.1 – Некоторые функции обработки строк

Номер функции	Функция
1	chunk_split – разбивает строку на фрагменты
2	echo – выводит одну или более строк
3	explode – разбивает строку с помощью разделителя
4	implode – объединяет элементы массива в строку
5	lcfirst – преобразует первый символ строки в нижний регистр
6	ltrim – удаляет пробелы или другие символы из начала строки
7	print – выводит строку
8	printf – выводит отформатированную строку
9	rtrim – удаляет пробелы или другие символы из конца строки
10	similar_text – вычисляет степень похожести двух строк
11	sprintf – возвращает отформатированную строку
12	sscanf – разбирает строку в соответствии с заданным форматом
13	stripos – возвращает позицию первого вхождения подстроки без учета регистра
14	strlen – возвращает длину строки
15	str_pad – дополняет строку другой строкой до заданной длины
16	str_replace – заменяет все вхождения строки поиска на строку замены
17	str_shuffle – переставляет символы в строке случайным образом
18	str_split – преобразует строку в массив
19	str_word_count – возвращает информацию о словах, входящих в строку
20	strip_tags – удаляет теги html и PHP из строки
21	strpbrk – ищет в строке любой символ из заданного набора
22	strpos – возвращает позицию первого вхождения подстроки
23	strrchr – находит последнее вхождение символа в строке
24	strrev – переворачивает строку задом наперед
25	stripos – возвращает позицию первого вхождения строки без учета регистра
26	strlen – возвращает длину строки
27	strpbrk – ищет в строке любой символ из заданного набора
28	strpos – возвращает позицию первого вхождения подстроки
29	strrchr – находит последнее вхождение символа в строке
30	strrev – переворачивает строку задом наперед
31	strripos – возвращает позицию последнего вхождения строки без учета регистра
32	strrpos – возвращает позицию последнего вхождения подстроки в строке
33	strspn – возвращает длину участка в начале строки, соответствующего маске
34	strstr – находит первое вхождение подстроки
35	strtok – разбивает строку на токены
36	strtolower – преобразует строку в нижний регистр
37	strtoupper – преобразует строку в верхний регистр
38	strtr – преобразует заданные символы или заменяет подстроки
39	substr_count – возвращает число вхождений подстроки
40	substr_replace – заменяет часть строки
41	substr – возвращает подстроку
42	trim – удаляет пробелы или другие символы из начала и конца строки

Порядок выполнения работы.

1 Работа со строками:

– определить строку с использованием синтаксиса одинарных кавычек;

– определить строку с использованием синтаксиса двойных кавычек;

– определить строку с использованием ***heredoc***-синтаксиса;

– создать массив из трех-пяти элементов, вывести его с использованием echo, print, print_r, serialize и пояснить полученные результаты.

2 Составить программу на языке PHP с использованием функций, указанных в таблице 13.2, согласно варианту.

Таблица 13.2 – Варианты заданий

Номер варианта	Номер функции	Номер варианта	Номер функции	Номер варианта	Номер функции
1	1, 13, 25, 37	5	5, 17, 29, 41	9	9, 21, 33, 45
2	2, 14, 26, 38	6	6, 18, 30, 42	10	10, 22, 34, 46
3	3, 15, 27, 39	7	7, 19, 31, 43	11	11, 23, 35, 47
4	4, 16, 28, 40	8	8, 20, 32,44	12	12, 24, 36, 48

Контрольные вопросы

1 Каковы особенности строк, записанных в одинарных кавычках?
2 Как объединить элементы массива в строку?
3 Что понимается под ***heredoc***-синтаксисом?
4 Как удалить пробелы или другие символы из конца строки?
5 Как преобразовать строку в массив?
6 Что такое ***heredoc***-текст?
7 В чем особенности строк, записанных в двойных кавычках?
8 Как удалить пробелы из начала и конца строки?
9 Какая функция переворачивает строку задом наперед?
10 Какие функции преобразуют символы строки в верхний регистр?

14 Лабораторная работа № 14. Изучение операторов цикла языка PHP

В языке PHP существует несколько конструкций, позволяющих выполнять повторяющиеся действия в зависимости от условия. Это циклы ***while***, ***do ...while, foreach и for***.

while – это простой цикл. Он имеет две формы записи:

1) ***while*** (выражение) {блок_выполнения};

2) ***while*** (выражение): блок_выполнения ***endwhile***.

Циклы ***do..while*** похожи на циклы ***while***, но в них истинность выражения проверяется в конце цикла. Форма записи:

do {блок_выполнения} ***while*** (выражение).

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

Синтаксис цикла ***for***:

for (инициализирующие_команды; условие; команды_после_итерации) { тело_цикла; }

Цикл ***for*** начинает свою работу с выполнения инициализирующих_команд, которые выполняются только один раз. Затем проверяется условие_цикла и, если оно истинно (***true***), выполняется тело_цикла. После того как будет выполнен последний оператор тела, выполняются команды_после_итерации. Затем снова проверяется условие_цикла. Если оно истинно (***true***), выполняется ***тело_цикла*** и ***команды_после_итерации*** и т. д. Например:

```
<?php
for ($x=0; $x<10; $x++) echo $x;
?>                  // выводит: 0123456789
```

Если необходимо указать несколько команд, то их можно разделить запятыми, например:

```
<?php
for ($x=0, $y=0; $x<10; $x++, $y++) echo $x;
?>                  // Выводит 0123456789
```

Пример использования нескольких команд в цикле ***for***:

```
<?php
for($i=0,$j=0,$k="Точки"; $i<10; $j++,$i+=$j) { $k=$k."."; echo $k; }
                    // Выводит Точки.Точки..Точки...Точки....
?>
```

Цикл ***for*** имеет альтернативный синтаксис:

```
for(инициализирующие_команды; условие; команды_после_итерации);
операторы;
endfor;
```

Цикл ***foreach*** перебора массивов имеет синтаксис:

```
foreach (массив as $ключ=>$значение)
  команды;
```

Пример цикла ***foreach***:

```
<?php
$names["Иванов"] = "Андрей";          $names["Петров"] = "Борис";
$names["Волков"] = "Сергей";          $names["Макаров"] = "Федор";
foreach ($names as $key => $value) {
```

```
echo "<b>$value $key</b><br>";
  }
?>
```

выводит: Андрей Иванов
Борис Петров
Сергей Волков
Федор Макаров

Порядок выполнения работы.

Написать и отладить скрипт, выполняющий действия, указанные в таблице 14.1, согласно варианту.

Таблица 14.1 – Варианты заданий

Номер вари-анта	С помощью цикла
1	Найти сумму корней чисел от 1 до 15. Результат округлить до двух знаков в дробной части
2	Найти сумму тех чисел от 1 до 100, которые делятся на 7
3	Создать строку из шести символов, состоящую из случайных чисел от 1 до 9
4	Дан массив с числами, найти сумму квадратов его элементов
5	Дан массив с числами, найти корень из суммы квадратов элементов этого массива, а результат округлить в меньшую сторону до целых
6	Дан массив с числами, найти сумму тех его чисел, которые больше 0 и меньше 10
7	Заполнить двумерный массив, содержащий 10 подмассивов, случайными числами от 1 до 10. В каждом подмассиве должно быть по 10 элементов
8	Преобразовать строку 'var_text_hello' в 'varTextHello'. Скрипт должен работать с любыми строками такого типа
9	Дан массив с произвольными числами. Сделать так, чтобы элемент в массиве повторился количество раз, соответствующее его значению. Например, [1, 3, 2, 4] должен превратиться в [1, 3, 3, 3, 2, 2, 4, 4, 4, 4]
10	Дана строка, удалить из этой строки четные символы
11	Дана строка, поменять ее первый символ на второй и наоборот, третий на четвертый и наоборот, пятый на шестой и наоборот и т. д., т. е. из строки '12345678' необходимо сформировать строку '21436587'
12	Написать скрипт, который проверяет, являются ли заданные числа простыми, т. е. делящимися только на единицу и сами на себя

Контрольные вопросы

1 Какие операторы цикла Вы знаете?
2 Какие формы записи конструкции ***while*** Вы знаете?
3 Прокомментируйте работу цикла ***do..while***.
4 Для чего используется конструкция ***for***?
5 Какие формы записи конструкции ***for*** Вы знаете?
6 Для чего используется оператор ***break*** в цикле ***for***?
7 Прокомментируйте работу конструкции ***foreach***.

8 Какие конструкции цикла используют для работы с массивами?
9 Как прервать выполнение цикла?
10 Для чего используется оператор ***break***?

15 Лабораторная работа № 15. Изучение приемов работы с массивами на языке PHP

В языке PHP в одном массиве допускается хранение переменных различных типов, а также массивов и объектов. Для обращения к элементу массива используется его индекс (ключ).

PHP поддерживает работу с индексными и ассоциативными массивами, индексами которых являются строки.

Для обращения к элементам индексных массивов используются числовые индексы, а ассоциативных – строковые.

Для создания массивов можно использовать конструкцию array() или способ приведения скалярной переменной типа ***int***, ***float***, ***string*** или ***boolean*** к типу ***array***, а также специализированные функции:

array([…]) – создает массив из значений, переданных конструкции в качестве параметров ***array_fill($start_index***, ***$num***, ***$value***), которая возвращает массив, содержащий ***$num*** элементов, имеющих значение ***$value***. Нумерация индексов при этом начинается со значения ***$start_index***;

range($low, $high [, $step]) – создает массив со значениями из интервала от $iow до $high и шагом $step;

explode($delimiter, $str [, $limit) – возвращает массив из строк, каждая из которых соответствует фрагменту исходной строки ***$str***, находящемуся между разделителем, определяемым аргументом ***$delimiter***. Необязательный параметр ***$limit*** определяет максимальное количество элементов в массиве, при этом последний элемент будет содержать остаток строки ***$str***.

В качестве элементов массива могут выступать другие массивы, в этом случае говорят о многомерных массивах. Массивы можно создавать, обращаясь к элементам или используя вложенные конструкции ***array()***. Для вывода массива используется функция ***print_r()***.

Работу с ассоциативными массивами удобно выполнять с использованием специализированного оператора цикла ***foreach***.

При манипуляции с массивами и их элементами часто возникает необходимость определения количества элементов в массиве. Для решения этой задачи используются следующие функции:

count($array [,$mode]) – возвращает количество элементов массива $array. Если $mode принимает значение count_recursive, функция рекурсивно обходит многомерный массив, в противном случае подсчитывается количество элементов только на текущем уровне;

sizeof() – синоним для функции ***count()***;

array_count_values($input) – подсчитывает количество уникальных значений среди элементов массива и возвращает ассоциативный массив, ключами которого являются значения массива, а значениями – количество их вхождений в массив ***$input***.

Порядок выполнения работы.

Написать и отладить скрипт, выполняющий действия, указанные в таблице 15.1, согласно варианту.

Таблица 15.1 – Варианты заданий

Номер варианта	Задание
1	В массиве из *n* строк проверить, начинается ли каждая строка символом «*», а строки без «*» перенести в другой массив
2	В массиве из *n* строк проверить, содержит ли *k*-я строка символ @. Если не содержит, то вставить этот символ в конец строки
3	В массиве строк удалить все html-теги, заключенные в скобки < >
4	В массив случайным образом поместить строки, содержащие «цитата дня». Для выбора строки из массива случайным образом можно использовать функции ***Shuffle***(***array arr***) или ***arrayrand***(***array arr***, ***int num***)
5	Создать многомерный массив: Факультет, Курс, Группа, Студенты. Вывести список студентов в алфавитном порядке
6	Создать многомерный массив: Факультет, Кафедра, Преподаватель, Ученое_звание. Вывести список преподавателей в алфавитном порядке
7	Создать двухмерный массив, в первой строке которого записаны номера и названия месяцев года в произвольном порядке. Во второй строке рассортировать месяцы года в алфавитном порядке, а в третьей – в порядке возрастания номера месяца
8	Заполнить элементы квадратной матрицы возрастающими числами, начиная с единицы по спирали, начиная с элемента [1, 1] по часовой стрелке
9	Заполнить элементы квадратной матрицы возрастающими числами, начиная с единицы по спирали, начиная с элемента [*n*, *n*] против часовой стрелки
10	Перемножить две числовые матрицы, размеры и значения матриц выбрать самостоятельно
11	Найти максимальную сумму диагональных элементов квадратной матрицы, размер и значения матрицы выбрать самостоятельно
12	Найти суммы элементов двух квадратных матриц и выбрать матрицу с большей суммой, размеры и значения матриц выбрать самостоятельно

Контрольные вопросы

1 Что такое ассоциированный массив?
2 Допускается ли хранение в одном массиве значений разных типов?
3 Какие конструкции используются для создания массивов?
4 Что такое индексный массив?
5 Как создать двухмерный массив?
6 Для чего используется функция ***Shuffle(array arr)***?

7 Может ли быть элементом массива другой массив?
8 Прокомментируйте назначение функции ***arrayrand(array arr, int num)***.
9 Можно ли хранить в массиве другие объекты?
10 Какие специализированные функции создания массивов Вы знаете?

16 Лабораторная работа № 16. Изучение условных операторов PHP

К условным операторам языка PHP относятся операторы ***if*** с расширениями ***else***, ***elseif***, а также конструкция ***switch***, позволяющая проверить условие и выполнить в зависимости от его истинности определенные действия.

Структура оператора ***if***:

```
if (выражение)
        инструкция
```

Здесь выражение – это любое правильное PHP-выражение, которое в процессе обработки скрипта преобразуется к логическому типу. Если в результате преобразования значение выражения истинно, то выполняется блок_выполнения, в противном случае блок_выполнения игнорируется. Если блок_выполнения содержит несколько команд, то он заключается в фигурные скобки, например

```
<?php
if(200 > 100) {
      echo 'Условие ';
      echo 'верно.';
   } else {
      echo 'Условие ';
      echo 'неверно.';
   }
?>
```

Структура оператора ***switch***:

```
switch (выражение или переменная) {
        case значение1:
           блок_действий1
                break;
        case значение2:
           блок_действий2
                break;
        ...
        default:
           блок_действий_по_умолчанию
 }
```

Для конструкции ***switch***, как и для ***if***, возможен альтернативный синтаксис, где открывающая их фигурная скобка заменяется двоеточием, а закрывающая – ***endswitch*** и ***endif*** соответственно.

Порядок выполнения работы.

Написать и отладить скрипт, выполняющий действия, указанные в таблице 16.1, согласно варианту.

Таблица 16.1 – Варианты заданий

Номер варианта	Задание
1	В массиве ***$a*** хранятся целые числа и слова. Определить количество и сумму положительных, отрицательных чисел, а также количество слов в массиве
2	В массиве ***$x*** хранятся целые и вещественные числа. Определить произведение всех целых и сумму вещественных чисел, больших 25,5
3	Переменная ***$s*** содержит строку, состоящую из положительных и отрицательных чисел и слов, разделенных запятыми. Сформировать массив чисел и определить среднее их абсолютных значений, а также строку слов
4	Переменная ***$min*** содержит строку разделенных пробелом чисел из интервала от 0 до 59. Определить, сколько чисел попадает в какую четверть часа, а также суммы чисел в каждой четверти
5	Переменная ***$lang*** может принимать два значения: ***ru*** и ***en***. Если она имеет значение ***ru***, то в переменную ***$arr*** записать массив дней недели на русском языке, а если ***en*** – то на английском. Задачу решить с использованием ***if***, ***switch-case*** и многомерного массива без ***if*** и ***switch***
6	В переменной ***$a*** содержатся целые числа, разделенные пробелами. Если выделяемые значения равны или меньше пяти, то в переменную ***$b*** записать их сумму, а если больше или равны девяти, то разность
7	Если *i*-я переменная массива ***$a*** больше двух и меньше девяти или переменная ***$b*** больше или равна шести и меньше девяти, то вывести ***Верно***, в противном случае – ***Неверно***.
8	В массиве ***$day*** содержатся числа из интервала от единицы до 31. Определить, в какую декаду месяца попадают эти числа
9	Переменная ***$month*** содержит числа из интервала от единицы до 12. Определить, в какую пору года попадает этот месяц: зима, весна, лето или осень
10	В переменной ***$year*** хранится номер года. Определить, является ли он високосным. Год является високосным, если он делится на 4, но при этом не делится на 100 либо делится на 400
11	Дана строка с цифрами, например, ***12345***. Проверить, является ли первым символом этой строки цифра ***1***, ***2*** или ***3***. Если является, то вывести ***да***, в противном случае вывести ***нет***
12	Дана строка из шести цифр. Проверить, равняется ли сумма первых трех цифр сумме вторых трех цифр. Если это так, вывести ***да***, в противном случае – ***нет***

Контрольные вопросы

1 Какие операторы в языке PHP относятся к условным?
2 Какие расширения используются в операторе ***if***?

3 Для чего используется оператор ***switch***?
4 Прокомментируйте структуру оператора ***switch***.
5 Для чего используется расширение ***else*** в операторе ***if***?
6 Для чего используется расширение ***elseif*** в операторе ***if***?
7 Что понимается под альтернативным синтаксисом оператора ***if***?
8 Для чего используется ***break*** в операторе ***switch***?
9 Как оформить несколько команд в блоке выполнения оператора ***if***?
10 Что понимается под альтернативным синтаксисом оператора ***switch***?

17 Лабораторная работа № 17. Изучение технологии работы с функциями PHP

В PHP существуют две основные формы функций: встроенные и пользовательские.

Полный список встроенных PHP-функций можно просмотреть в окне редактора кода, нажав кнопку ***Поиск*** в правой колонке при пустой строке поиска ***PHP-поиск***.

Среди встроенных имеется большое количество функций для работы со строками, например, функция замены символов ***str_replace( )***. Пример функции для замены в строке ***abracadabra*** символа ***a*** на ***o*** может иметь вид:

```
<?php
$string = 'abracadabra';
echo str_replace('a', 'o', $string);
?>
```

Пользовательская функция создается с помощью оператора ***function***, после которого через пробел указываются ее имя и круглые скобки, которые могут быть пустыми либо содержать принимаемые функцией аргументы.

После объявления функции в фигурных скобках записывается ее код. Общий синтаксис пользовательской функции имеет вид:

```
function имяФункции (аргумент1, аргумент2, …) {
   действие;
   return результат;
}
```

Пользовательская функция может иметь нуль и более аргументов и возвращает результат, который присваивается указанной после ***return*** переменной, или не возвращает результат, а, например, выводит информацию на экран. Оператор ***return*** прекращает выполнение функции и может быть размещен в любом ее месте.

Вызывается функция по ее имени, после которого обязательны круглые скобки, даже если они пустые.

В функцию параметры можно передать по ссылке, указывая знак ***&*** перед именем аргумента:

```
function func(&$x)
```

Если передать переменную ***$a*** в функцию по ссылке ***&$x***, то изменяя значение переменной ***$x*** внутри функции, будет изменяться исходное передаваемое значение. То есть значение переменной ***$a***, передаваемое функции, присваивается переменной ***$x*** и если изменить значение переменной ***$x*** внутри функции, то оно изменится и вне ее:

```
function plus1(&$x) {
    $x = $x + 5;
  }
$a = 3;
Plus1($a);
echo $a;
```

Результат: ***$a*** = 8.

Если же переменную ***$a*** передать в функцию по значению, то при вызове функции значение переменной ***$a*** будет просто скопировано в локальную переменную ***$x*** функции ***plus2($x)***. Внутри этой функции значение переменной ***$x*** увеличивается на ***5***. Но эта локальная переменная ***$x*** вне функции перестает существовать и ее влияние на переменную ***$a*** не проявляется:

```
function plus2($x) {
    $x = $x + 5;
  }
$a = 3;
Plus2($a);
echo $a;
```

Результат: ***$a*** = 3

Рекомендуется по возможности использовать функции, называемые чистыми, в которых аргументы передаются по значению, т. к. передача их по ссылкам часто приводит к запутанности кода.

В PHP допускается использование динамических функций. Это означает, что если некоторой переменной присвоено имя функции, то с этой переменной можно обращаться точно так же, как с самой функцией.

В функциях допускается также использование глобальных переменных, созданных с помощью инструкции ***global*** вне функции.

Порядок выполнения работы.

Написать и отладить скрипт, выполняющий действия, указанные в таблице 17.1, согласно варианту.

Таблица 17.1 – Варианты заданий

Номер вари-анта	Задание
1	Создать две пользовательские функции, первая из которых принимает два аргумента и возвращает их произведение, а вторая формирует массив делителей заданного числа – чисел, на которые оно делится без остатка. Вызвать функцию и результат вывести на экран
2	Создать пользовательскую функцию, принимающую два аргумента по ссылке и один по значению, которая присваивает переменным числовые значения. Вызвать функцию и вывести на экран произведение всех переменных
3	Создать пользовательскую функцию, присваивающую двум переменным численные значения и выполняющую их произведение. Вызвать функцию, передав ей в качестве аргументов сначала значения двух переменных, затем одной и, наконец, вообще без аргументов. Результат вывести на экран
4	Создать пользовательскую функцию, принимающую аргументы в массив переменной длины и выводящую их на экран. Для доступа к элементам массива использовать цикл ***foreach***. Вызвать функцию, передав ей в качестве значения две строки и число
5	Создать пользовательскую функцию, вычисляющую корень из заданного четырехзначного числа и округляющую его в большую и меньшую стороны. В массив ***arr*** первым элементом записать заданное число, вторым – корень из этого числа, третьим – округление в меньшую сторону, четвертым – округление в большую сторону
6	Заполнить массив 50 случайными числами от 25,4 до 98,7. Вычислить квадратный корень из каждого числа, округлить его до целого в большую и меньшую сторону и результаты округления записать в ассоциативный массив с ключами **max** и **min**
7	Дано целое двухзначное число, равное значению 50 + Ваш номер в журнале. Создать функцию, определяющую делители этого числа, т. е. числа, на которые оно делится без остатка. Сформировать массив делителей заданного числа
8	Создать пользовательскую функцию, которая принимает два целочисленных аргумента и возвращает сумму их квадратов. Вызвать функцию, передав ей в качестве аргументов сначала два целых числа, а затем одно из них в виде строки
9	Создать пользовательскую функцию, которая принимает два целочисленных аргумента, и возвращать их произведение. Присвоить переменной имя созданной функции, обратиться к функции через переменную, и вывести на экран полученный результат
10	Создать переменную и присвоить ей целое число. Создать еще одну переменную и присвоить ей анонимную функцию, наследующую эту переменную и выводящую на экран ее инкрементированное значение. Выполнить вызов функции, затем изменить значение внешней переменной и снова вызвать функцию. Изменить скрипт, задав наследование переменной по ссылке
11	Создать функцию, вычисляющую квадратный корень из заданного числа. Результат округлить в большую и меньшую сторону до целого, а результаты округления записать в ассоциативный массив с ключами ***floor*** и ***ceil***
12	Создать функцию, определяющую, сколько разных элементов массива [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] нужно сложить, чтобы сумма получилась больше 10

Контрольные вопросы

1 Как просмотреть подробное описание конкретной PHP-функции?
2 Какие две основные формы PHP-функций Вы знаете?
3 Как создаются пользовательские PHP-функции?
4 Что записывается в фигурных скобках в PHP-функции пользователя?
5 Как посмотреть полный список встроенных PHP-функций?
6 Прокомментируйте общий синтаксис PHP-функции.
7 Как вызвать PHP-функцию?
8 Как использовать глобальные переменные в PHP-функциях?
9 Как вернуть результат пользовательской PHP-функции?
10 Что такое динамические PHP-функции?

Список литературы

1 **Никольский, А. П.** JavaScript на примерах / А. П. Никольский. – Москва: Наука и техника, 2017. – 274 с.

2 **Симпсон, К.** ES6 & Beyond/ES6 и не только / К. Симпсон. – Санкт-Петербург: Питер, 2017. – 336 с.

3 **Кузнецов, М.** Самоучитель PHP 7 / М. Кузнецов, И. Симдянов. – Санкт-Петербург: БХВ-Петербург, 2018. – 450 с.

4 **Скляр, Д.** Изучаем PHP 7. Руководство по созданию интерактивных веб-сайтов: пер. с англ. / Д. Скляр. – Санкт-Петербург: Альфа-книга, 2017. – 464 с.: ил.