МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ

Государственное бюджетное профессиональное образовательное учреждение

Московской области

«Воскресенский колледж»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

**ПМ 05.** **Основы** **WEB - дизайна**

**Наименование специальности**

54.01.20 Графический дизайнер

**Квалификация выпускника**

графический дизайнер

АННОТАЦИЯ

Методические рекомендации разработаны для студентов по профессии 54.01.20 Графический дизайнер по ПМ 05. Основы WEB - дизайна и предназначены для выявления уровня сформированности общих и профессиональных компетенции в рамках модуля МДК 05.01. Web – дизайн МДК 05.02. WEB – программирование

Методическая разработка содержит теоретический и практический материал по МДК 05.01. Web – дизайн МДК 05.02. WEB – программирование

# ВВЕДЕНИЕ

Основное назначение дисциплины МДК 05.01. Web – дизайн МДК 05.02. WEB – программирование в средних профессиональных образовательных организациях состоит в формировании у студентов общих и профессиональных компетенций: выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам, осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности, планировать и реализовывать собственное профессиональное и личностное развитие, использовать информационные технологии в профессиональной деятельности, планировать предпринимательскую деятельность в профессиональной сфере, осуществлять сбор, систематизацию и анализ данных необходимых для разработки технического задания дизайн- продукта, выполнять процедуру согласования (утверждения) с заказчиком, планировать выполнение работ по разработке дизайн-макета на основе технического задания, определять потребности в программных продуктах, материалах и оборудовании при разработке дизайн-макета на основе технического задания, проводить мастер-классы, семинары и консультации по современным технологиям в области графического дизайна. Содержание дисциплины предусматривает повторение и систематизацию знаний, полученных в средней общеобразовательной школе, формирование общих и профессиональных компетенций.

Практическое занятие – это форма организации учебного процесса, предполагающая выполнение обучающимися заданий самостоятельно и под руководством преподавателя. Дидактическая цель практических работ – формирование у обучающихся профессиональных и практических умений, необходимых для изучения последующих учебных дисциплин, а также подготовка к применению этих умений в профессиональной деятельности.

Практические занятия предполагают работу:

* по созданию целостного миропонимания;
* по расширению мировоззренческого кругозора;
* по объяснению объективной реальности и предельных оснований человеческих действий в системе логических категорий и законов.

Структура практических занятий включает в себя теоретические вопросы по изучаемым темам, упражнения на усвоение основных понятий web-технологий в дизайне, использование информационных технологий в области графического дизайна.

Программа учебной дисциплины может быть использована при проведении дополнительных занятий на курсах по дисциплине «Web-технологии в дизайне» и подготовительных курсах для поступающих.

**МДК 05.01. Основы WEB- дизайна**

**Практическая работа № 1**

**Объектная модель браузера (BOM)**

Цель: познакомиться с объектной моделью браузера, изучить объекты и их свойства, позволяющие управлять поведением браузера из JavaScript.

**Теория**

С помощью объектной модели браузера (Browser Object Model) можно управлять поведением браузера из JavaScript.

Для каждого открытого окна браузер создает объект **Window.** Все объекты JavaScript, включая переменные и функции, хранятся в объекте **window**. Методы объекта: **open()** открывает новое окно, **close()** закрывает окно, **alert()** выводит окно оповещения.

Window включает перечисленные ниже объекты.

Объект **Navigator** позволяет определить, какой браузер использует пользователь. Следующий пример показывает, как с помощью свойства **userAgent** можно вывести на страницу информацию о браузере пользователя

document.write('Информация о браузере:'+navigator.userAgent);

С помощью свойства **appVersion** можно узнать версию браузера, свойства **appName** – название браузера, свойства **appCodeName** – кодовое название браузера, свойства **platform** – ОС, которую использует пользователь. Метод **javaEnabled()** объекта **Navigator** дает возможность проверить, включена ли поддержка Java в браузере пользователя или нет. Метод вернет true, если поддержка включена, и false, если нет.

Объект **Screen** содержит информацию об экране пользователя. Свойство **width** определяет ширину экрана пользователя, свойство **height** – высоту. С помощью свойства **colorDepth** можно узнать глубину цвета (измеряется в битах на пиксель).

Объект **History** содержит список URL, которые были посещены в данном окне браузера. С помощью свойства **length** можно узнать количество посещенных URL, хранящихся в списке.

Объект **Location** позволяет узнать информацию о URL текущего документа. Свойство **href** хранит URL текущего документа целиком. С помощью свойства **pathnam** можно узнать путь к загруженному документу. Свойство **host** содержит имя домена загруженного документа. Метода **assign()** позволяет загрузить новый документ в данное окно браузера (изменить текущий URL на желаемый).

**Задания к лабораторной работе**

***Задание 1.*** Создать кнопки, открывающие и закрывающие окна с произвольными именами. Создать кнопки, выводящие в каждое из окон какой-нибудь текст, включающий имя окна. Проверить работу скрипта в разных браузерах.

***Задание 2.*** Узнать:

1. всю информацию о браузере пользователя;
2. версию браузера;
3. название браузера;
4. кодовое название браузера;
5. ОС, которую использует пользователь;
6. включена ли поддержка Java в браузере**;**
7. ширину и высоту экрана;
8. глубину цвета;
9. URL, которые были посещены в данном окне браузера;
10. URL текущего документа;
11. путь к загруженному документу;
12. имя домена загруженного документа.

Результаты вывести в таблице:

Таблица

|  |  |
| --- | --- |
| свойство | значение |
|  |  |

**Практическая работа № 2**

Арифметические, логические операторы, операторы сравнения в JavaScript.

**Цель: изучить возможности ввода и вывода данных; особенности использования арифметических, логических операторов, операторов сравнения, операторов условного перехода.**

Теория

Ввод и вывод данных.

**JavaScript был создан в 1995 году как инструмент, предоставляющий веб-дизайнерам возможности программирования. Все современные браузеры имеют поддержку JavaScript. JavaScript встраивается прямо в веб-страницы и исполняется браузером во время их загрузки/**

**JavaScript добавляется на веб-страницы с помощью тэга <script>. В следующем примере на страницу выводится текст «Я изучаю JavaScript».**

<script type="text/javascript">

document.write("Я изучаю JavaScript.");

</script>

**JavaScript код также может храниться во внешнем текстовом файле с расширением .js и подключаться к страницам с помощью тэга <script> следующим образом:**

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

**JavaScript чувствителен к регистру букв.**

**В JavaScript существует два вида комментариев: короткие и длинные. Все, что находится после символа // до конца строки, будет являться коротким комментарием. Многострочные комментарии начинаются с /\* и заканчиваются \*/.**

**В JavaScript предусмотрены три стандартных метода для ввода и вывода данных: **alert()**, **prompt()** и **confirm().****

**Метод alert()** **позволяет выводить диалоговое окно (окно оповещения) с заданным сообщением и кнопкой ОК. Используется в случаях, когда необходимо, чтобы пользователь обязательно обратил внимание на определенную информацию.**

Метод **confirm()** позволяет вывести диалоговое окно (окно подтверждения) с сообщением и двумя кнопками – ОК и Отмена (Cancel). Используется в случаях, когда необходимо, чтобы пользователь подтвердил или отклонил что-либо. Если пользователь щелкнул на кнопке ОК, то возвращается значение true; если щелкнул на кнопке Отмена, то возвращается значение false.

Метод **prompt()** позволяет вывести на экран диалоговое окно (окно запроса) с сообщением, а также с текстовым полем, в которое пользователь может ввести данные. Используются в случаях, когда от пользователя необходимо получить определенную информацию. В этом окне предусмотрены две кнопки: ОК и Отмена (Cancel). В отличие от методов alert() и confirm() данный метод принимает два параметра: сообщение и значение, которое должно появиться в текстовом поле ввода данных по умолчанию.

Пример применения **методов для ввода и вывода данных:**

**<html>**

**<head>**

**<script type='text/javascript'>**

**function fun1() {**

**alert('Я окно оповещения');**

**}**

**function fun2() {**

**confirm('Я окно подтверждения');**

**}**

**function fun3() {**

**x=prompt('Введите Ваше имя:', 'Имя');**

**document.write('Здравствуйте, '+ x);**

**}**

**</script>**

**</head>**

**<body>**

**<input type='button' value='Окно оповещения' onclick='fun1()' />**

**<input type='button' value='Окно подтверждения' onclick='fun2()' />**

**<input type='button' value='Окно запроса' onclick='fun3()' />**

**</body>**

**</html>**

Типы данных.

Типы данных представлены в таблице

Таблица

Типы данных в Java Script

|  |  |  |
| --- | --- | --- |
| Тип данных | Примеры | Описание значений |
| Строковый или символьный (string) | "Привет"  "д.т. 123-4567" | Последовательность символов, заключенная в кавычки, двойные или одинарные |
| Числовой (number) | 3.14 -567 +2.5 | Число, последовательность цифр, перед которой может быть указан знак числа (+ или перед положительными числами не обязательно ставить знак «+»; целая и дробная части чисел разделяются точкой. Число записывается без кавычек |
| Логический (булевский, boolean) | true false | true (истина, да) или false (ложь, нет); возможны только два значения |
| Null |  | Отсутствие какого бы то ни было значения |
| Объект (object) |  | Программный объект, определяемый своими свойствами. В частности, массив также является объектом |
| Функция (function) |  | Определение функции – программного кода, выполнение которого может возвращать некоторое значение |

**Арифметические операторы.**

Арифметические операторы используются для выполнения арифметических операций над переменными или значениями.

Таблица

Арифметические операторы

|  |  |  |
| --- | --- | --- |
| Оператор | Название | Пример |
| + | Сложение | X + Y |
| - | Вычитание | X – Y |
| \* | Умножение | X \* Y |
| / | Деление | X / Y |
| % | Деление по модулю | X % Y |
| ++ | Увеличение на 1 | Х++ |
| -- | Уменьшение на 1 | Y-- |

Чтобы уменьшить размер кода, можно использовать сокращенную запись арифметических операций. Например, вместо x= x + y писать x+ = y.

**Операторы сравнения**

Операторы сравнения представлены в таблице

Таблица

**Операторы сравнения**

|  |  |  |
| --- | --- | --- |
| Оператор | Название | Пример |
| == | Равно | X==Y |
| != | Не равно | X!=Y |
| > | Больше, чем | X>Y |
| >= | Больше или равно | X>=Y |
| < | Меньше, чем | X<Y |
| <= | Меньше или равно | Х <= Y |

Оператор «равно» записывается с помощью двух символов без пробелов между ними.

Логические операторы

Логические данные, обычно получаемые с помощью элементарных выражений, содержащих операторы сравнения, можно объединять в более сложные выражения. Для этого используются логические (булевские) операторы: И (&&), ИЛИ (||), НЕ (!). Выражения с логическими операторами возвращают значение true или false.

Операторы условного перехода

Оператор условного перехода **if** позволяет реализовать структуру условного выражения если ..., то ..., иначе ...

Пример оператора if перехода:

if (sr\_ball>=5)

{document.write("Стипендия есть");

}

else

{document.write("Стипендии нету");

}

В фигурных скобках располагается блок кода – несколько выражений. Если в блоке используется одна команда, то фигурные скобки можно не писать.

Оператор **Switch** (переключатель) удобно использовать, если требуется проверить несколько условий, которые не являются взаимоисключающими.

Пример оператора switch:

*s*witch (sr\_ball)

{

case 5: document.write("Стипендия минимальная")

case 6: document.write("Стипендия с коэффициентом 1,2")

case 9: document.write("Стипендия с коэффициентом 1,4")

}

**Задания к лабораторной работе**

***Задание 1.*** Опробовать работу с модальными окнами **alert**, **prompt**, **confirm**.

Разработать свой диалог с использованием этих команд. Пример диалога:

* вывести приветствие в окне alert “Вас приветствует учебный центр”;
* затем в окне prompt вывести сообщение «Введите имя», а в окне alert вывести сообщение «Добро пожаловать на наши курсы, *имя*»;
* в окне confirm вывести сообщение "Хотите стать Web-дизайнером?" с альтернативными ответами в окнах alert, если TRUE, то "Учите стили CSS и JavaScript!", если FALSE то "Упускаете время!".

***Задание 2.*** Использование оператора + с числовыми и строковыми переменными.

1. сложить два любых числа (например, 10 и 5). Результат вывести на страницу;
2. сложить две строки (например, “10” и “5”). Результат вывести на страницу;
3. сложить число и строку (например, 22 и “5”). Результат вывести на страницу;
4. сложить строку и число (например, “22” и 5). Результат вывести на страницу;
5. вывод – *Результатом сложения строки и числа всегда будет …* – вывести в окно.

***Задание 3.*** Найти значение 2-х любых арифметических выражений, записанных на JS с использованием арифметических операций (например, 35y-25x)/5+232 и  8\*y/x+5\*x/y – 43)\*6). Значения *x и y* задать произвольно, но подобрать такие, чтоб значение одного выражения было намного больше второго.  Найти остаток от деления значения одного выражения на значение другого. Вывести результаты на страницу, а также в отдельное окно.

***Задание 4.*** Продемонстрировать использование операторов сравнения (==, !=, >, >=, <, <=) и логических операторов (!, &&, ||).

Введите любое число. Если оно меньше 20 или больше 40 и не равно 15 и делится без остатка на 5, то вывести сообщение «Правильное значение», иначе «Не правильное значение».

Придумайте свой пример с логическими операторами.

***Задание 5.*** Конструкция if..elseif..else.

Ввести 2 числа с клавиатуры. Сравнить их и вывести сообщение на страницу, например “*А больше Б*”, используя конструкцию if..elseif..else.

Написать свой пример с использованием альтернативного синтаксиса конструкции if..else(условие)?команды1:команды2

***Задание 6.*** Определить, какой сегодня день недели (использовать конструкцию switch case).

***Задание 7***\****.*** Придумать пример, демонстрирующий обработку исключений с использованием конструкции try … catch.

**Практическая работа № 3**

**Функции. Операторы цикла.Объекты Math, Number**

Цель: научиться создавать пользовательские функции на JS, использовать в программах операторы цикла; изучить особенности использования встроенных объектов **Math** и **Number**.

**Теория**

Функции

Функция представляет собой подпрограмму, которую можно вызвать для выполнения, обратившись к ней по имени. Взаимодействие функции с внешней программой, из которой она была вызвана, происходит путем передачи функции параметров и приема от нее результата вычислений. Функция в JavaScript может и не требовать параметров, а также ничего не возвращать.

В JavaScript есть встроенные функции, которые можно использовать в программах, но код которых нельзя редактировать или посмотреть. Пользователь также может создать свои функции для решения конкретных задач – пользовательские.

Описание функции имеет следующий синтаксис:

function имя\_функции(параметры){

код

}

Функция **messageWrite()** в примере ниже будет выполнена только после нажатия на кнопку. В примере используется событие **onclick**.

<html>

<head>

<script type='text/javascript'>

// Функция выводит текст на страницу

function messageWrite() {

document.write('Текст выводится на страницу с помощью JavaScript!');

}

</script>

</head>

<body>

<input type='button' value='Нажми на меня' onclick='messageWrite()' />

</body>

</html>

Операторы цикла

Цикл – это блок команд, который может повторно выполняться, пока определенное условие не будет выполнено.

JavaScript поддерживает 3 вида циклов: **for**, **while**, **do..while**.

**Оператор for**

Цикл for исполняет блок команд, пока заданное условие является истинным. Например,

for (i=1;i<=30;i++) {

document.write (i+'<br />');

}

**Оператор while**

Цикл while выполняет блок кода, пока заданное условие истинно. Напрмер,

var i=1;

while (i<=30) {

document.write (i+'<br />');

i++;

}

**Оператор do-while**

Цикл do..while часто называют циклом с постусловием, потому что в отличие от предыдущих циклов он вначале исполняет блок команд и только потом проверяет заданное условие. Если условие истинно, блок команд выполняется еще раз, если условие ложно, цикл завершает исполнение.

В отличие от оператора while в операторе do-while код выполняется хотя бы один раз, независимо от условия. В примере код выполнится один раз, так как условие ложно.

var i=20;

do {

document.write('Я студент (от лат. studens — усердно работающий, занимающийся) БГТУ');

}

while (i<=3);

**Объекты**

Объекты представляют собой программные единицы, обладающие некоторыми свойствами. Об объекте можно судить по значениям его свойств и описанию того, как он функционирует.

Встроенные объекты имеют фиксированные названия и свойства. Все свойства этих объектов разделяют на два вида: просто свойства и методы. Свойства аналогичны обычным переменным. Они имеют имена и значения. Некоторые свойства объектов доступны только для чтения, их значения нельзя изменять. Другие свойства доступны и для записи – их значения можно изменять с помощью оператора присваивания. Методы аналогичны функциям, они могут иметь параметры или не иметь их.

Таким образом, объект можно понимать как некоторый контейнер, содержащий переменные-свойства и функции-методы.

Для разработчиков веб-сайтов особенно важны объекты String (обработка строк), Array (массивы), Math (математические формулы и константы) и Date (работа с датами).

Объект Math

Встроенный объект **Math** позволяет производить математические операции. Чтобы обращаться к свойствам и методам этого объекта его не нужно предварительно создавать (в отличие от остальных встроенных объектов JavaScript).

Свойства объекта **Math** содержат значения часто используемых математических констант. В примере выводятся на страницу некоторые константы.

//Выведем на страницу число Пи

document.write(Math.PI + '<br />');

//Выведем экспоненту

document.write(Math.E + '<br />');

//Выведем натуральный логарифм 10

document.write(Math.LN10 + '<br />');

//Выведем квадратный корень 2

document.write(Math.SQRT2);

С помощью методов объекта можно производить над числами различные математические операции. Ниже перечислены методы **Math**:

* аbc(число) – возвращает модуль (абсолютное значение) числа;
* асоs(число) – возвращает арккосинус числа;
* asin(число) – возвращает арксинус числа;
* atan(число) – возвращает арктангенс числа;
* atan2(x, у) – возвращает угол в полярных координатах точки;
* ceil(число) – округляет число вверх до ближайшего целого;
* cos(число) – возвращает косинус числа;
* ехр(число) – возвращает число **е** в степени число;
* floor(число) – округляет число вниз до ближайшего целого;
* log(число) – возвращает натуральный логарифм числа;
* max(число1, число2) – возвращает большее из 2 чисел;
* min(число1, число2) – возвращает меньшее из 2 чисел;
* pow(число1, число2) – возвращает число1 в степени число2;
* random () – возвращает случайное число между 0 и 1;
* round(число) – округляет число до ближайшего целого;
* sin(число) – возвращает синус числа;
* sqrt(число) – возвращает квадратный корень из числа;
* tan(число) – возвращает тангенс числа.

В примере демонстрируется применение методов объекта **Math**.Все результаты выводятся на страницу.

<html>

<body>

<script type='text/javascript'>

//Округлим число 25.34 до ближайшего целого

document.write(Math.round(25.34) + '<br />');

//Округлим число 25.88 до ближайшего целого

document.write(Math.round(25.88));

//Выберем из чисел 10 68 35 12 44 максимальное

document.write(Math.max(10,68,35,12,44) + '<br />');

//Выберем из чисел 10 68 35 12 44 минимальное

document.write(Math.min(10,68,35,12,44));

//Возведем 5 в -1 степень

document.write(Math.pow(5,-1));

//Сгенерируем случайное число между 0 и 1

document.write(Math.random() + '<br />');

//Сгенерируем случайное число между 0 и 100 и округлим

document.write(Math.round(Math.random()\*100));

</script>

</body>

</html>

Объект Number (Число)

В JavaScript числа могут быть двух типов: целые и с плавающей точкой. Числа можно создавать обычным образом с помощью переменных и оператора присваивания, не прибегая к объекту Number. Однако этот объект обладает некоторыми полезными свойствами и методами, которые иногда могут пригодиться.

Пример создания объекта Number:

x = new Number(‘34.21’).

Методы Number:

* **toExponential(количество)** – представляет число в экспоненциальной форме, параметр *количество* – целое число, определяющее, сколько цифр после точки следует указывать.
* **toFixed(количество)** – представляет число в форме с фиксированным количеством цифр после точки, параметр *количество* – целое число, определяющее, сколько цифр после точки следует указывать.
* **toPrecision(точность)** – представляет число с заданным общим количеством значащих цифр. Параметр *точность* – целое число, определяющее, сколько всего цифр, до и после точки, следует указывать.
* **toString([основание])** – возвращает строковое представление числа в системе счисления с указанным основанием. Если параметр не указан, имеется в виду десятеричная система счисления. Этот метод имеют все объекты.

Например, следующий код выведет на страницу число 3.422e+1:

x = new Number(34.215);

x=x.toExponential(3);

document.write(x).

**Задания к лабораторной работе**

***Задание 1.*** Все выполненные задания из предыдущей работы оформить в виде функций. Создать кнопки и запускать задание на выполнение щелчком по соответствующей кнопке. Поработать над оформлением результатов вывода (цвет шрифта, кегль, гарнитура).

***Задание 2.*** Вывести таблицу умножения, **а** и **b** ввести с клавиатуры. Использовать оператор цикла **for.**

Таблица

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | … | b |
| 2 | 4 | … | 2\*b |
| … |  |  |  |
| a | a\*2 | … | a\*b |

Применить к ячейкам таблицы свойства форматирования.

***Задание 3.*** Найти площадь круга и длину окружности, радиус меняется от **а** до **b** с шагом 0,3. Результаты округлить и вывести в таблице. Использовать оператор цикла **do**-**while.**

Таблица

|  |  |  |
| --- | --- | --- |
| Радиус | Площадь круга | Длина окружности |
| a |  |  |
| a+0,3 |  |  |
| … |  |  |
| b |  |  |

Применить к ячейкам таблицы свойства форматирования.

***Задание 4.*** Создать 2 объекта-числа (Number) – дробное и целое. Применить к каждому из них методы: **toExponentia**l, **toFixed**, **toPrecision**, **toString**. Результаты проанализировать, представить в таблице в следующем виде:

Таблица.

|  |  |  |  |
| --- | --- | --- | --- |
| Число | Метод | Результат | Описание метода |
| 127.18 | toString(16) | 7f.2e147ae147b | Строковое представление числа в 16-ричной системе счисления |
|  |  |  |  |

**Практическая работа №4**

**Встроенные объекты JavaScript**

Цель: изучить встроенные объекты JavaScript Array, String, Date, их свойства и методы, научиться их правильно применять в своих скриптах.

**Теория**

Объект Array (Массив)

Массив представляет собой упорядоченный набор данных. Его удобно представить себе в виде одностолбцовой таблицы, содержащей некоторое количество строк. В ячейках такой таблицы могут находиться данные любого типа, в том числе и массивы.

Можно создать массивы тремя разными способами.

Первый cпособ:

spec = new Array();

spec[0] = ИСиТ;

spec[1] = "ПОИТ";

spec[2] = "ПОИБМС";

spec[3] = "ДЭВИ";

Второй способ:

var spec = new Array("ИСиТ","ПОИТ","ПОИБМС","ДЭВИ");

Третий способ:

var spec = [("ИСиТ","ПОИТ","ПОИБМС","ДЭВИ")];

Нумерация индексов в массивах начинается не с 1, а с 0. С помощью свойства **length** можно узнать количество элементов в массиве.

Методы Array:

* **concat()** объединяет два и более массива в один;
* **рор()** – удаляет последний элемент массива и возвращает его значение;
* **push(значение|объект)** – добавляет к массиву указанное значение в качестве последнего элемента и возвращает новую длину массива;
* **shift()** – удаляет первый элемент массива и возвращает его значение;
* **sliсe(индекс1 [, индекс2])** – создает массив из элементов исходного массива с индексами указанного диапазона;
* **sort([функция\_сортировки])** – сортирует (упорядочивает) элементы массива с помощью функции сравнения.

**Объект String (Строка)**

Объект String (строковый объект) используется для хранения и обработки текстовой информации.

Примеры создания:

mystring = new String ("Привет!") – первый способ;

mystring = "Привет!" – второй способ.

С помощью свойства **length** можно узнать длину строки:

document.write(mystring.length)

С помощью метода **toUpperCase()** можно перевести все символы текста в верхний регистр, а с помощью **toLowerCase()** – в нижний.

Метод **concat()** позволяет объединить две и более строки и вывести результат на страницу:

str='Кафедра';

document.write(str.concat('ИСиТ'));

Метод **replace()** позволяет заменить одно произвольное слово в строке на другое:

document.write(str.replace('ИСиТ','ПОИТ')).

**Объект Date (Дата)**

Объект **Date** позволяет производить различные операции с датой и временем. Некоторые методы объекта представлены в таблице.

Таблица

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| [getDate()](http://www.wisdomweb.ru/JSd/datdat.php) | Возвращает день месяца (может принимать значения от 1-31) заданной даты. |
| [getDay()](http://www.wisdomweb.ru/JSd/datday.php) | Возвращает день недели (может принимать значения от 0-6, причем 0-Воскресенье, а 6-Суббота) заданной даты. |
| [getFullYear()](http://www.wisdomweb.ru/JSd/datyear.php) | Возвращает год (4 числа, например 2017) заданной даты. |
| [getHours()](http://www.wisdomweb.ru/JSd/dathours.php) | Возвращает час (может принимать значения от 0-23) заданной даты. |
| [getMiliseconds()](http://www.wisdomweb.ru/JSd/datmil.php) | Возвращает миллисекунду (может принимать значения от 0-999) заданной даты. |
| [getMinutes()](http://www.wisdomweb.ru/JSd/datmin.php) | Возвращает минуту (может принимать значения от 0-59) заданной даты. |
| [getMonth()](http://www.wisdomweb.ru/JSd/datmon.php) | Возвращает месяц (может принимать значения от 0-11) заданной даты. |
| [getSeconds()](http://www.wisdomweb.ru/JSd/datsec.php) | Возвращает секунду (может принимать значения от 0-59) заданной даты. |
| [toDateString()](http://www.wisdomweb.ru/JSd/todatestring.php) | Преобразует часть объект, содержащую дату, в строку. |
| [toString()](http://www.wisdomweb.ru/JSd/tostring.php) | Преобразует объект в строку. |
| [toTimeString()](http://www.wisdomweb.ru/JSd/totimestring.php) | Преобразует часть объекта, содержащую время, в строку. |

После того, как объект создан, можно с помощью доступных методов производить над ним различные операции.

Пример.

//Создадим объект Date

x=new Date();

//Извлечем день месяца и выведем результат на страницу

document.write(x.getDate());

//Извлечем год из объекта x и выведем результат на страницу

document.write(x.getFullYear());

**Задания к лабораторной работе**

***Задание 1.*** Сформировать массив (объект Array), элементами которого являются значения выражений. Для вычисления выражений использовать объект Math. Найти максимальный и минимальный элементы массива и их номера.

1. 
2. 
3. 
4. 

***Задание 2.*** Задан массив с элементами, представляющими методы объектов Array и Math (pow, pop, push, shift, round, floor, sline, sort). Получить из него 2 массива, в один записать методы объекта Array, в другой – методы объекта Math.

Добавить в начало одного массива и в конец другого еще по одному методу соответствующих объектов.

Вывести исходный массив, полученные массивы и их длину (количество элементов).

***Задание 3.*** Создать объект **String** – строку текста (свои Фамилия Имя Отчество), в которой присутствуют строчные и прописные буквы. Узнать ее длину.

Перевести все символы строки в верхний регистр, а затем в нижний. Соединить полученные строки. Заменить свои Фамилия Имя Отчество на ФИО.

Вывести исходную и полученные строки на страницу.

***Задание 4.*** Использовать объект **Date**. Вывести на страницу таблицу с составляющими текущей даты и времени в виде:

Таблица .

|  |  |
| --- | --- |
| Год | 2017 |
| Месяц |  |
| День |  |
| Час |  |
| Минуты |  |
| Секунды |  |

Применить к ячейкам таблицы свойства форматирования.

**Практическая работа №5**

**Работа с формами**

Цель: научиться создавать на веб-странице формы, изучить возможные элементы формы и их свойства, а также способы доступа к элементам формы.

**Теория**

HTML формы могут содержать такие элементы ввода как:

* текстовые поля;
* флажки;
* радио-кнопки;
* кнопки отправления и др.

Формы также могут содержать списки выбора, многострочные текстовые поля, метки и др.

Для создания формы в HTML используется тег **<form>**.

Элементы ввода используются для приема пользовательских данных. Они отличаются друг от друга в зависимости от значения атрибута **type**.

Текстовое поле **<input type="text"/>** определяет однострочное текстовое поле, в которое пользователь может вводить различную информацию.

Поле пароля **<input type="password"/>** определяет поле для ввода пароля.

Флажок **<input type="checkbox"/>** позволяет пользователям выбирать несколько пунктов с предварительно заполненной информацией из группы.

Радио-кнопка **<input type="radio"/>** позволяет выбрать только один пункт с предварительно заполненной информацией из группы.

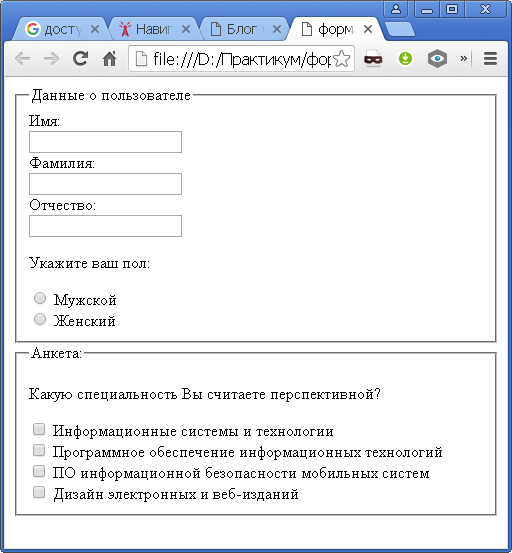
Кнопка отправления **<input type="submit"/>** предназначена дл отправления на сервер введенных данных. Адрес, на который будут пересылаться данные формы, указывается в атрибуте тэга **form** – **action**. Если данный атрибут отсутствует, данные будут отправлены на текущую страницу.

Текстовая область **<textarea>** позволяет ввести многострочный текст.

Список **<select>** создает выпадающий списков, элементы которого определяются с помощью тэга **<option>.** С помощью атрибута **multiple** можно указать, что в выпадающем списке могут быть выбраны одновременно несколько элементов.

Заголовки в формах **<fieldset>** позволяют сгруппировать желаемую часть формы и затем с помощью тэга **<legend>** установить желаемое заглавие.

Пример формы:



Представленная выше форма создана следующим кодом:

<form>

<fieldset>

<legend>Данные о пользователе</legend>

Имя: <br /><input type='text' name='firstname' /><br />

Фамилия:<br /> <input type='text' name='lastname' /><br />

Отчество: <br /><input type='text' name='lastname' /><br />

<p>Укажите ваш пол:</p>

<input type='radio' name='s' value='m' /> Мужской<br />

<input type='radio' name='s' value='f' /> Женский

</fieldset>

<fieldset>

<legend> Анкета: </legend>

<p>Какую специальность Вы считаете перспективной?</p>

<input type='checkbox' name='space' value='1' />

Информационные системы и технологии<br />

<input type='checkbox' name='space' value='2' />

Программное обеспечение информационных технологий <br />

<input type='checkbox' name='space' value='3' />

ПО информационной безопасности мобильных систем <br />

<input type='checkbox' name='space' value='3' />

Дизайн электронных и веб-изданий <br />

</fieldset>

</form>

**Доступ к элементам формы.**

С помощью JavaScript можно отправить данные формы на сервер, очистить ее, а также получить доступ к любому элементу формы для изменения его значений.

Существует несколько способов обращения к формам с использованием объектной модели.

При создании формы автоматически создается массив **forms.** Для доступа к форме требуется указать номер элемента или имя формы, заданное параметром **name**.

<html> <body>

<form name=data>  
...  
</form>

<script>

alert(document.forms.length) // количество форм на странице

alert(document.forms[0].name) // имя первой формы

alert(document.forms.data.length) // количество элементов в форме с именем data

alert(document.forms["data"].length) // То же самое

</script>  
</body> </html>

Нумерация элементов массива всегда начинается с нуля, поэтому обращение к первой форме будет **document.forms[0]**, ко второй – **document.forms[1]**.

Второй способ – через семейство **all.** Обращение к форме происходит как к элементу массива с именем, совпадающим с именем формы – **document.all["data"]** или напрямую – **document.all.data.**

<html> <body>

<form name=data>

...

</form>

<script>

alert(document.all["data"].length)

alert(document.all.data.name)

</script>  
</body> </html>

Обращение к элементам формы осуществляется посредством семейства **elements** или напрямую по имени элемента.

<html><body>

<form name=data> <input type=text name=userName value="Введите ваше имя"> </form>

<script>

alert(document.forms.data.elements.length) // Общее число элементов в форме

alert(document.forms[0].elements[0].value) // Значение первого элемента

alert(document.forms["data"].userName.value) // Значение элемента с именем userName

</script>

</body></html>

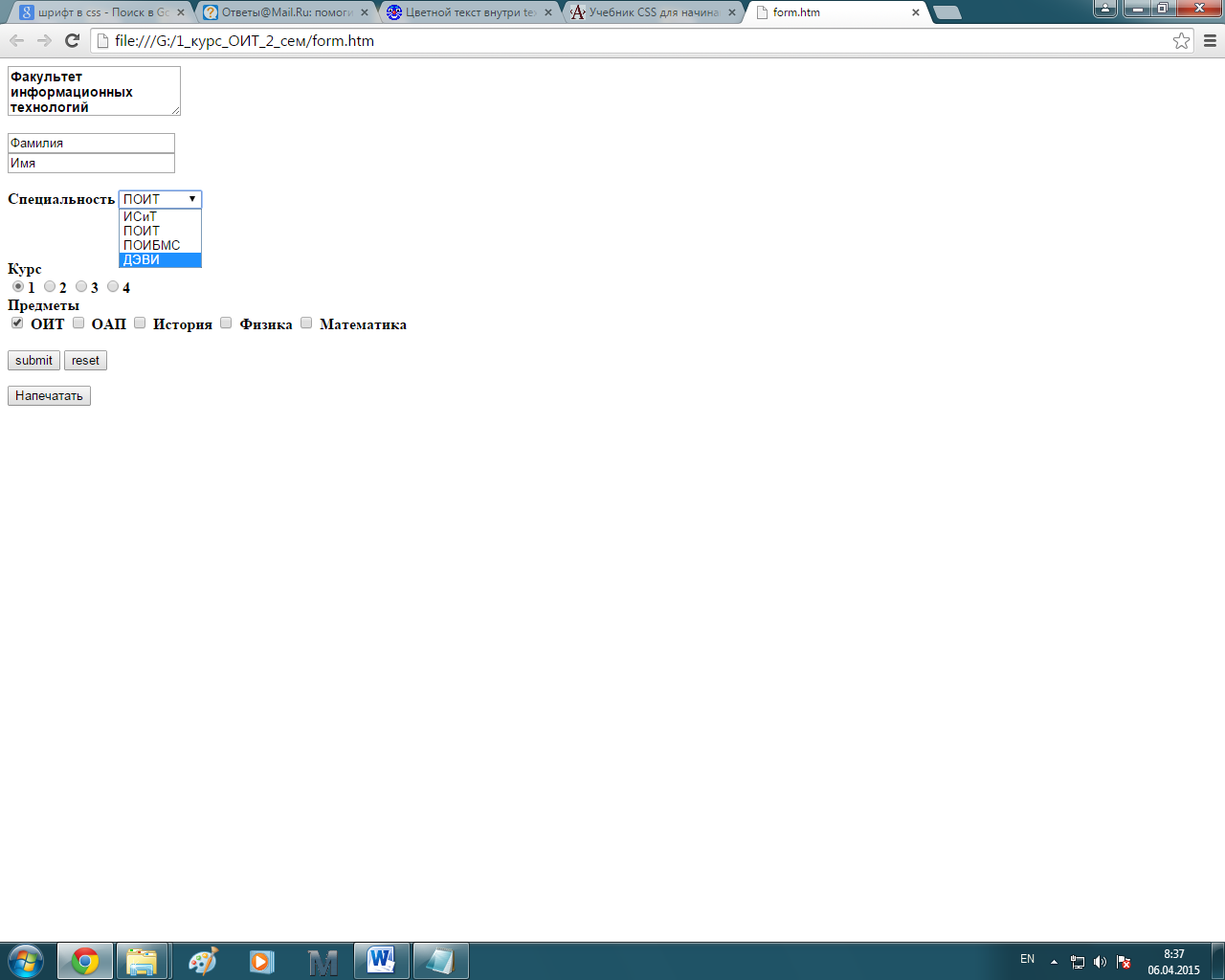
Нумерация массива **elements**, как и в случае с семейством forms ведется с нуля, поэтому обращение к первому элементу формы будет **elements[0].** Для большого количество данных в форме значения элементов лучше получать по их имени.

Доступ к элементам формы можно также осуществить по их ID с помощью метода **document.getElementById()**.

Задания к лабораторной работе

***Задание 1.*** Проанализировать код для создания формы, представленный выше. Изучить правила создания элементов формы.

***Задание 2.*** Создать форму, содержащую элементы (текстовая область, 2 текстовых поля, список, 4 радиокнопки, 5 флажков, командные кнопки). Вид формы представлен ниже.



***Задание 3.*** Для кнопки «Напечатать» написать функцию, выводящую на страницу примерно такой текст (подчеркнутые фразы взять из элементов формы, используя обращение к ним по имени, а также по номеру элемента):

Факультет информационных технологий

Студент Фамилия специальность ИСиТ курс 1 должен сдавать следующие предметы:

* ОИТ
* История
* Математика

***Задание 4.*** Создать еще одну форму с одним элементом – списком. Заполнить список предметами, выбранными на первой форме.

Можно придумать свою форму с разными элементами. Это приветствуется.

**События. Динамические эффекты на JS**

Цель: научиться создавать динамические эффекты средствами JavaScript.

**Теория**

События – это функции, которые могут быть привязаны к элементам HTML страниц.

Код событий выполнится только после того, как произойдет их активирующее действие. Разные типы событий имеют разные активирующие действия.

Дополнительную информацию о событиях, которые произошли, содержит объект события **events**, то есть в объекте events хранятся атрибуты событий. Например, атрибут **screenX** позволяет узнать горизонтальные координаты указателя мыши во время вызова события относительно границ экрана, атрибут **screenY** – вертикальные координаты, **target** позволяет узнать элемент, который вызвал событие, **type** – позволяет узнать имя события и т. д.

В любой момент времени существует не более одного объекта события. Все инициированные события заносятся операционной системой в буфер и выполняются последовательно в том порядке, в каком они туда попали.

Примеры активирующих действий JavaScript:

* щелчок мыши (событие **onclick**);
* нажатие клавиши (**onkeypress**);
* отправление формы (**onsubmit**);
* наведение курсора мыши на элемент (**onmouseover**);
* выведение курсора мыши за пределы границ элемента (**onmouseout**);
* полная загрузка страницы или картинки (**onload**);
* изменение содержимого элемента, например содержимого текстового поля формы (**onchange**).

Метод **getElementById(значение\_ID)** позволяет обратиться к любому элементу по значению его идентификатора – значению атрибута ID. Если несколько элементов документа имеют одинаковый ID, метод возвращает первый элемент с указанным значением ID.

Свойство **srcElement** возвращает ссылку на объект элемента HTML-документа, который инициировал событие. При получении такой ссылки можно узнать или изменить значения свойств этого объекта и применить к нему любой из его методов.

Следующий код на JS позволяет при щелчке кнопкой мыши на картинке увеличивать изображение, а затем при щелчке на увеличенном изображении его уменьшать.

<script>

var pic\_sm = new Array("mcat1.jpg", "mcat2.jpg", "mcat3.jpg")

var pic\_bg = new Array("cat1.jpg", "cat2.jpg", "cat3.jpg")

var pic\_flag = new Array(pic\_sm.length) *//* массив флагов

/\* Формирование строки тегов изображений \*/

var xstr = ""

for (i=0; i<pic\_sm.length; i++)

xstr += '<img id="img'+i+'" src="'+pic\_sm[i]+'" onclick="imgchange()">'

document.write(xstr)

function imgchange() {

var xid=event.srcElement.id //id изображения, на котором был щелчок

var n=parseInt(xid.substr(3)) // выделяем номер элемента

if (pic\_flag[n]) document.all[xid].src=pic\_sm[n]

else document.all[xid].src=pic\_bg[n]

pic\_flag[n] = !pic\_flag[n]

}

</script>

**Задания к лабораторной работе**

***Задание 1.*** Создать программу на JS, изменяющую цвет слова в тексте при наведении курсора мыши (событие Onmouseover, свойство style.color).

***Задание 2.*** Создать программу на JS, изменяющую размер шрифта слова щелчком мыши (событие Onclick, свойство style.fontSize).

***Задание 3.*** Создать программу на JS замены картинки на другую при щелчке мышью с помощью метода getElementById и свойства src.

***Задание 4.*** Создать программу на JS, заменяющую текст изображением, с использованием метода getElementById и свойства innerHTML.

***Задание 5.*** Создать эффекты увеличения размера рисунка (свойство width) при наведении на него курсора мыши и возврата к первоначальному размеру при уходе курсора с рисунка (событие onmouseout)

***Задание 6.*** Нарисовать цветную рамку произвольного стиля вокруг любого абзаца при двойном щелчке (событие dblclick) на абзац.

**Практическая работа №6**

**Движение объектов и графика на JavaScript**

Цель: изучить возможности управления процессами во времени: методы **setInterval()** и **setTimeout()**; познакомиться с приемами реализации перемещения и движения объектов; научиться отображать графики.

**Теория**

**Управление процессами во времени**

Для организации постоянного периодического (через заданный интервал времени) выполнения некоторого выражения или функции служит метод **setInterval()** объекта window. Этот метод имеет следующий синтаксис:

setInterval (выражение, период [, язык]).

Первый параметр – строка, содержащая выражение (обычно вызов функции). Второй параметр – целое число, указывающее временную задержку в миллисекундах. Третий, необязательный параметр, указывает язык, на котором написано выражение; по умолчанию – JavaScript. Метод **setInterval()** возвращает некоторое целое число – идентификатор временного интервала, который может быть использован в дальнейшем для прекращения процесса.

В следующем примере функция myfunc() выполняется периодически через 0,5 с.

setInterval("myfunc()", 500)

Для остановки запущенного временного процесса служит метод **clearInterval(идентификатор)**, который принимает в качестве параметра целочисленный идентификатор, возвращаемый соответствующим методом setInterval(), например:

var myproc = setInterval("myfunc () , 100")

if (confirm("Прервать процесс ?"))

clearInterval(myproc)

Чтобы выполнить выражение с некоторой временной задержкой, используется метод **setTimeout()** объекта window, который имеет следующий синтаксис:

setTimeout(выражение, задержка [, язык]).

Первый параметр – строка, содержащая выражение (обычно вызов функции). Второй параметр – целое число, указывающее временную задержку в миллисекундах. Третий, необязательный параметр указывает язык, на котором написано выражение; по умолчанию – JavaScript. Метод setTimeout() возвращает некоторое целое число – идентификатор временного интервала, который может быть использован в дальнейшем для отмены задержки выполнения процесса.

Для отмены задержки процесса, запущенного с помощью метода setTimeout(), используется метод **clearInterval(идентификатор)**, который принимает в качестве параметра целочисленный идентификатор, возвращаемый соответствующим методом setTimeout.

В следующем HTML-документе имеются две кнопки. Щелчок на кнопке Пуск открывает через 5 с новое окно и загружает в него документ mypage.htm. Это действие можно отменить с помощью кнопки Отмена, если щелкнуть на ней, пока окно еще не открыто:

<HTML>

<BUTTON ID="start">Пуск</BUTTON>

<BUTTON ID="stop">Отмена</BUTTON>

<SCRIPT>

var myproc

function start.onclick(){

myproc = setTimeout ("window.open('mypage.htm')", 5000)

}

function stop.onclick(){

clearTimeout(myproc)

}

</SCRIPT>

</HTML>

**Перемещение графических объектов**

Перемещать мышью изображения можно различными способами. Изучим один из них: пользователь пытается перетащить мышью изображение; затем он должен отпустить кнопку мыши и переместить указатель в нужное место; остановившись в нужном месте, пользователь отпускает кнопку мыши или щелкает ею, чтобы прекратить перемещение изображения. Код перемещения изображения мышью:

<НТМL>

<НЕАD><TITLE>Перемещаемое изображение</TITLE></НЕАD>

<BODY id = "mybody">

<IMG ID="myimg" SRC = "pict.gif ondragstart = "drag()" style = "position:absolute; top:10; left:10">

</BODY>

<SCRIPT>

flag = false // нельзя перемещать

var id\_img = ""

function drag() {

flag = mag

id\_img = event.srcElement.id

}

function mybody.onmousemove(){

if (flag){ // если можно перемещать

document.all[id\_img].style.top = event.clientY

document.all[id\_img].style.left = event.clientX } }

function mybody.onmouseup(){

flag = false // нельзя перемещать

}

</SCRIPT>

</HTML>

Здесь функция drag(), обрабатывающая событие ondragstart (попытка перетаскивания), устанавливает переменную-триггер flag и выясняет, кто инициатор события. Значение переменной flag позволяет определить, можно или нельзя перемещать элемент. В данном примере инициатором события может быть только один элемент. События onmousemove (перемещение указателя мыши) и onmouseup (кнопка мыши отпущена) получает не изображение, а объект тела документа mybody.

**Перемещение текстовых областей**

Рассмотрим пример HTML-документа, в котором можно переме­щать текстовые области, созданные с помощью тегов <TEXTAREA>. Размеры области и шрифта текста определяются в ней в зависимости от значения вертикальной координаты.

<HTML>

<НЕАD><ТITLE>Перемещаемые текстовые облас-ти</ТITLE></НЕАD>

<BODY id = "mybody" background = "blue.gif">

<TEXTAREA ID="tl" ondblclick=""drag()" STYLE = "position:absolute;

top:10; left: 10; fon-size: large"> Это – первый текст</TEXTAREA>

<TEXTAREA ID="t2" ondblclick=""drag()" STYLE = "position:absolute;

top:100; left:150"> Это – второй текст </TEXTAREA>

<TEXTAREA ID="t3" ondblclick=""drag()" STYLE = "position:absolute; top:150; left:250"> Это - третий текст </ТЕXTAREA>

</BODY>

<SCRIPT>

resizetext() { // установка размеров текстовых областей

var flag = false }

var id\_img = ""

function drag() {

flag = !flag

id\_img = event.srcElement.id //id элемента, который надо перемещать

function mybody.onmousemove() {

if (flag){

document.all[id\_img].style.top = event.clientY

document.all[id\_img].style.left = event.clientX

resizetext() }} // установка размеров текстовых областей

function mybody.onmouseup() {

flag = false }

function resizetext () { // установка размеров областей

var у, size, idimg, idtext

for (i =0 ; i < document.all.length; i++) {

if (document.all[i].tagName=='TEXTAREA') {

idtext = document.all[i].id

у = parseInt(document.all[idtext].style.top)

size = Math.min(y, 800)

size = Math.max(size, 60)

document.all[idtext].style.width = size

document.all[idtext].style.height = 0.8\*size

document.all[idtext].style.zIndex = у

document.all[idtext].style.fontSize = Math.max(2, y/10)

</SCRIPT>

</HTML>

Схема сценария, осуществляющего непрерывное перемещение видимого элемента документа, имеет следующий вид:

Function init\_move() { // инициализация движения

... // подготовка к запуску функции move()

setInterval("move()", задержка)

function move(){

.../\* изменение координат top и left стиля перемещаемого элемента \*/

// вызов функции для перемещения элемента

Cоздаются две функции: **init\_move(),** которая осуществляет подготовку исходных данных и вызывает метод **setInterval()** с указанием в качестве первого параметра имени второй функции **move()** в кавычках, и **move()**, которая изменяет координаты элемента. Поскольку метод setInterval() вызывает функцию move() периодически через заданное количество миллисекунд, то координаты элемента изменяются постоянно. При этом создается эффект движения. Скорость и плавность движения зависят от величин приращения координат (в функции move()) и временной задержки (второго параметра метода setInterval()). Чтобы начать перемещение элемента надо вызвать первую функцию init\_move().

**Остановка движения**

Сценарий с запуском и остановкой движения может выглядеть, например, следующим образом:

function init\_move(xid, dx, dy) {

var prmstr = "'"+xid+" ,"+dx+" ,"+dy // строка параметров для move()

prmstr = "move(" +prmstr+ ")"

idjnove = setInterval (prmstr , 200) /\* сохраняем идентификатор движения \*/

}

function move(xid, dx , dy) {

у = parseInt(document.all[xid].style.top)

x = parseInt(document.all[xid].style.left)

document.all.myimg.style.top = у+dy

document.all.myimg.style.left = x+dx

if (parseInt(document.all[xid].style.left) > 350) /\* остановка по условию \*/

}

clearInterval(id move)

init\_move(“myimg”, 10, 5) // начинаем движение

В этом примере движение остановится, как только горизонтальная координата элемента превысит 350 пикселов.

**Движение по произвольной кривой**

Рассмотрим задачу организации движения видимого элемента по произвольной кривой, заданной выражением с одной переменной. Функция движения в качестве параметров будет иметь два выражения, которые описывают изменения вертикальной и горизонтальной координат элемента. Эти выражения будут содержать одну переменную, которую мы обозначим через х – строчной латинской буквой. Переменную х можно интерпретировать как независимый параметр движения (например, время). С помощью встроенной функции eval() можно вычислить значения этих выражений при конкретном значении переменной х и присвоить их параметрам left и top таблицы стилей перемещаемого элемента. Функция (пусть это будет move()), которая все это выполняет, передается в качестве первого параметра методу setInterval(), который периодически вызывает ее через заданный интервал времени.

Функция инициализации движения **curvemove()** принимает три строковых параметра (ID перемещаемого элемента, выражение для вертикальной координаты и выражение для горизонтальной координаты) и один числовой параметр (период времени, через который координаты элемента пересчитываются). Ниже приводятся определения функций curvemove() и move():

function curvemove(xid, уехрr, xexpr, ztime) {

/\* Движение по произвольной кривой.\*/

if (!xid) return null

if (!yexpr) yexpr = "x"

if (!xexpr) xexpr = "x"

if (!ztime) ztime = 100 // интервал времени, мс

x = 0 /\* глобальная переменная, входящая в выражения уехрr и хехрr \*/

setInterval("move('"+xid+" ', + уехрг + " "+ xexpr +"' ztime)

}

function move(xid, yexpr, xexpr) {

x++

document.all[xid].style.top = eval(yexpr)

document.all[xid].style.left = eval(xexpr)

}

Параметры:

* хid – id движущегося объекта, строка;
* уехрr – выражение для вертикальной координаты;
* хехрr – выражение для горизонтальной координаты;
* ztime – интервал времени между вызовами функции move(), мс.

Чтобы сделать переменную х глобальной, не надо использовать ключевое слово var перед первым ее появлением в коде.

Ниже приведен пример HTML документа с движущимся изображением:

<НТМL>

<IMG ID = "myimg" SRC = "pict1.gif" STYLE = "position:absolute">

<SCRIPT>

function curvemove(xid, yexpr, xexpr, ztime) {

// код определения функции

}

function move(xid, yexpr, xexpr) {

// код определения функции

}

Curvemove()'myimg' , "100 + 50\*Math.sin(0.03\*x)" , "50 + х", 100)

</SCRIPT>

</HTML>

В этом примере изображение будет перемещаться по синусоиде с 50 пикселов и горизонтальной скоростью 10 пикселов в секунду. Начальные ординаты графического объекта равны 100 и 50 пикселов по вертикали и горизонтали соответственно.

**Рисование линий**

В JavaScript нет специальных встроенных средств для рисования произвольных линий. Для решения этой задачи ужно вывести на экран изображение размером 1x1 пиксел, залитое цветом, отличающимся от цвета фона. Это изображение следует разместить несколько раз в соответствии с координатами, которые задаются параметрами позиционирования top и left атрибута STYLE тега <IMG>. С помощью сценария можно сформировать строку, содержащую теги <IMG> с не­обходимыми атрибутами, а затем записать ее в документ методом write().

**Рисование прямой линии**

Для отображения точки в HTML можно использовать следующий тег:

<IMG SRC = "point.bmp" STYLE = "position:absolute; top:y; left:x">

Здесь point.bmp – имя графического файла, содержащего один пиксел; у, х – числа, указывающие положение графического файла в пикселах. Изображение точки размером 1x1 пиксел можно создать в любом графическом редакторе. Из соображений экономии его лучше всего сохранить в файле формата BMP, а не JPEG или GIF (при малых размерах изображения алгоритмы сжатия неэффективны).

Чтобы задать размеры отображения точки на экране, следует использовать атрибуты WIDTH и HEIGHT (ширина и высота):

<IMG SRC = "point.bmp" STYLE = "position: absolute; top:y; left:x" WIDTH=n HEIGHT=n>

Одинаковые значения атрибутов WIDTH и HEIGHT задают представление точки в виде квадрата размером n×n пикселов. При этом точка с исходными размерами 1x1 пиксел просто растягивается. Таким образом, имеется возможность задать отображаемые размеры (масштаб) одной точки, а, следовательно, и определить толщину линии.

Далее следует пример определения функции, которая рисует прямую линию с заданными координатами xl, yl, х2, у2 и толщиной линии n.

Function line(xl, yl, х2, у2 , n){

/\*xl, yl – начало линии, х2, у2 – конец, n – толщина \*/

var clinewidth = " WIDTH=" + n + "HEIGHT=" + n /\* строка для учета толщины \*/

var xstr = "" // строка тегов для записи в HTML-документ

var xstr0 = ' <IMG SRC="point.bmp"' + clinewidth + ' STYLE = "positlon:absolute; '

var k = (y2 – yl)/(x2 – xl) // коэффициент наклона линии

var x = xl // начальное значение координаты х

/\* Формирование строки, содержащей теги <IMG. . . >: \*/

while (x <= х2) {

xstr += xstr9 + 'top:' + (yl + k\* (x – xl)) + ': left:1 + x + '

x+ +

// запись в документ

**Рисование кривой линии**

Функция **curve()** для рисования кривой принимаtn следующие параметры: имя графического файла с изображением точки (может быть любое изображение), выражение, задающее кривую, координаты начала линии, количество точек линии, толщина линии и длина штриха (если потребуется штриховая линия). Далее следует код функции curve() для рисования кривых.

function curve(pict\_file,yexpr, x0, y0, t, n. s){

/\* pict\_file - имя графического файла

yexpr - выражение с переменной х

х0, у0 - координаты начала кривой

t - количество точек кривой (значений переменной х)

n - толщина линии

s - длина штриха и паузы \*/

if (!yexpr) return null

if (!pict\_file) pict\_file = "point.bmp"

if (!s) s = 0

if (!t) t = 0

var clinewidth = ''

if (!n)

clinewidth = 'WIDTH=' + n + 'HEIGHT=' + n

var x

xstrG = '<IMGSRO "' + pict\_file + '"'' + clinewidth + STYLE = "position:absolute;top: '

xstr = ""

var i = 0, draw = true

for(x = 0; x < t; x++) {

if (draw)

xstr += xstr0 + (y0 + eval(yexpr)) + '; left:' + (x0 + x)

if (i > s&&s > 0) {

draw = !draw

i = 0

}

i ++

}

document .write (xstr) // запись в документ

**Задания к лабораторной работе**

***Задание 1.*** Создать программу на JS, позволяющую перетащить мышью область текста и картинку.

***Задание 2.*** Создать программу на JS, организующую движение картинки по прямой линии.

***Задание 3.*** Создать программу на JS, организующую движение картинки по кривой линии и возврат ее в исходное состояние.

***Задание 4.*** Нарисовать график функции, выбрав функцию с помощью радио кнопки (y=x2, y=x3, y=sin(x), y=cos(x)).

***Задание 5\*.*** Реализовать выбор цвета графика функции с помощью списка.

**МДК 05.02. WEB -программирование**

**Практическая работа №1**

**«Основы»**

**Задачи обучающегося:**

* 1. Научиться оставлять комментарии в коде.
  2. Научиться именовать переменные
  3. Изучить использование операторов

**Опорные понятия:**Основы PHP.

**Планируемый результат:**

Студент должен

Уметь оставлять комментарии в коде. Знать назначение операторов.

Знать как использовать константы

**Необходимое оборудование:**ПК, IDENetBeans, OpenServer

**Алгоритм деятельности обучающегося:**

**Задача 1: Комментарии**

Создать файл php. В начале файла добавить 2 комментария (однострочный и многострочный) с именем автора и датой выполнения. Напечатать приветствие при помощи echo.

**Задача 2: Именование переменных**

* Придумайте и запишите имена переменных для таких сущностей: название телеканала, адрес производителя, цвет автомобиля, температура воды, модель телефона.
* Следуйте правилам именования переменных. Используйте lowerCamelCase для имен, составленных из двух и более слов.

**Задача 3: Операторы**

* Создайте 3 переменных с произвольными именами на свой выбор. Присвойте им значения 3, 5, 8 соответственно. Выведите значения этих переменных на страницу.
* Найдите сумму этих переменных и запишите ее в новую переменную. Выведите значение новой переменной.
* Найдите значение такого выражения: 2+6+2/5­1. Запишите его в переменную с именем

$result и выведите на страницу.

**Задача 4: Копирование значений переменных и копирование переменных по ссылке**

Будем использовать переменные $a, $b, $c, $d.

* Задайте значения 1 и 2 переменным $a, $b соответственно.
* Выведите значения переменных $a, $b.
* Скопируйте значение переменной $a в переменную $c.
* Скопируйте переменную $b в переменную $d по ссылке.
* Распечатайте значения переменных $c, $d.
* Присвойте переменным $a, $b значения 3 и 4 соответственно.
* Распечатайте переменнные $a, $b, $c, $d.
* Сделайте выводы относительно полученных результатов.

**Задача 5: Константы**

* Создайте 2 константы с произвольными именами со значениями 41 и 33.
* Найдите и выведите сумму этих констант.
* Попытайтесь переопределить одну из констант. Внимательно прочтите подсказку об ошибке в Вашей IDE (если поддерживается)

*Все задачи необходимо сделать в отдельных файлах (например: task1.php) Оформить отчет с листингом кода*

**«Типы данных»**

**Задачи обучающегося:**

1. Научиться определять типы переменных.
2. Научиться работать с переменными в строках
3. Научиться создавать простые массивы

**Опорные понятия:**Типы данных.

**Планируемый результат:**

Студент должен

Уметь выводить значения переменных Знать как работать с массивами.

**Необходимое оборудование:**ПК, IDENetBeans, OpenServer

**Алгоритм деятельности обучающегося:**

**Задача 1: Типы переменных**

Определите типы следующих переменных и выведите их на экран:

$a = 152;

$b = '152';

$c = 'London';

$d = array(152);

$e = 15.2;

$f = false;

$g = true;

**Задача 2: Работа со строками и переменными**

Написать код, который будет формировать строку и выводить ее на экран, подставляя в нее значения переменных $a и $b.

*Например:*

Дано: a = 10, b = 5.

Результат: 5 из 10ти студентов посетили лекцию.

*Условия*

Строки в PHP можно формировать используя одинарные ' и двойные " кавычки. Решите эту задачу двумя способами с применением одинарных и двойных кавычек.

*Подсказка:*

В случае использования двойных кавычек будет полезной возможность экранирования переменных при помощи { }.

**Задача 3: Работа со строками и переменными**

Определить три переменных со значениями:

“Доброе утро” “дамы”

“и господа”

* Вывести значения переменных в браузер.
* Сформировать строку "Доброе утро, дамы и господа" используя созданные переменные и комбинированный оператор склеивания.

**Задача 4: Работа с массивами**

* + Создать 2 простых массива с количеством элементов 5.
  + В первый массив добавить один элемент с индексом (!) element и произвольным значением.
  + Из второго массива удалить элемент с индексом 0. Используйте функцию unset();
  + Вывести на экран элементы под индексом 2 из первого и второго массива.
  + Вывести на экран содержимое массивов полностью.
  + Найти количество элементов в каждом массиве. Используйте функцию count(). Вывести результаты на экран.

*Все задачи необходимо сделать в отдельных файлах (например: task1.php) Оформить отчет с листингом кода*

**Практическая работа №2**

**«Условный оператор»**

**Задачи обучающегося:**

1. Изучить работу условного оператора

**Опорные понятия:**Условный оператор. **Планируемый результат:**

Студент должен

Уметь составлять программы для решения уравнений

Уметь составлять программы которая проверяет вхождение переменной в определенный диапазон

**Необходимое оборудование:**ПК, IDENetBeans, OpenServer

**Алгоритм деятельности обучающегося: Задача 1: Вхождение числа в диапазон**

Диапазон значений, определяется константами MIN = 10, MAX = 50.

Написать программу, которая проверяет вхождение переменной x в диапазон (больше MIN, меньше MAX).

Если х входит в диапазон, вывести сообщение "+", иначе вывести сообщение "­". Если х находится на границе диапазона (х = 10 или х = 50), вывести сообщение "+­".

*Подсказка:*

Подставляйте произвольные значения переменной х для проверки работы программы.

**Задача 2: Квадратное уравнение**

Решить квадратное уравнение.

Определить значения переменных $a, $b, $c произвольными числовыми значениями (с ними и работать дальше).

По формуле найти дискриминант (D).

Используя условный оператор (if, elseif, else) сравнить значение D c 0:

* Если D > 0, вычислить и напечатать 2 корня уравнения (x1, x2)
* Если D = 0, найти 1 корень (х)
* Если D < 0, вывести сообщение: "Нет корней" Подсказки:
  1. Вспомнить теорию и проверить работу своей программы можно на сайте <https://100formul.ru/kvadrat.php>
  2. Арифметические операторы мы уже знаем. Для нахождения квардратного корня используется

функция s qrt():

echosqrt(9); // Выведет "3"

**Задача 3: Равенство чисел**

Среди трех произвольно выбранных чисел найти среднее. Если среди чисел есть равные, вывести сообщение "Ошибка".

*Например:*

Дано: а = 2, b = 4, c = 3; Результат: "Среднее число: с = 3"

*Все задачи необходимо сделать в отдельных файлах (например: task1.php) Оформить отчет с листингом кода*

**«Циклы»**

**Задачи обучающегося:**

1. Изучитьработуциклов for, while, foreach

**Опорные понятия:**Циклы. **Планируемый результат:** Студент должен

Уметь решать задачи применяя разные типы циклов Уметь формировать меню для сайта на основе циклов

**Необходимое оборудование:**ПК, IDENetBeans, OpenServer

**Алгоритм деятельности обучающегося:**

**Задача 1: Сумма чисел**

Найти и напечатать сумму чисел от 1 до 25.

*Условие:*

Решить задачу двумя способами: используя цикл for и цикл while.

**Задача 2: Квадраты чисел**

Найти и вывести все квадраты натуральных чисел (1,2,3, ...), не превосходящие данного числа N.

Число N задаем самостоятельно.

**Задача 3: Меню на сайте**

Цель задачи ­ сформировать html­код списка для отображению меню на сайте.

Сформируйте массив (10 элементов) со строками вида: Кнопка 10

Кнопка 9

Кнопка 8

… Кнопка 1

Отсортируйте массив в обратном порядке любым способом. Если возникают сложности, Получить такой результат:

Кнопка 1

Кнопка 2

… Кнопка 10

При помощи echo, операторов склеивания и цикла foreach получить и вывести на экрантакой html­код:

<ul>

<li><a href="#">Кнопка 1</a></li>

<li><a href="#">Кнопка 2</a></li>

<li><a href="#">Кнопка 3</a></li>

...

<li><a href="#">Кнопка 10</a></li>

</ul>

*Все задачи необходимо сделать в отдельных файлах (например: task1.php) Оформить отчет с листингом кода*

**Практическая работа №3**

**«Пользовательские функции»**

**Задачи обучающегося:**

1. Научиться писать пользовательские функции исходя из поставленной задачи

**Опорные понятия:**Пользовательские функции.

**Планируемый результат:**

Студент должен

Уметь написать функцию под необходимую задачу

**Необходимое оборудование:**ПК, IDENetBeans, OpenServer

**Алгоритм деятельности обучающегося: Задача 1: Информация о товарах в корзине**

Написать функцию для получения итогов (общей информации) о корзине покупок. Функция принимает массив c информацией о выбранных товарах примерно такого вида:

$products = array(

array('name' => 'Телевизор', 'price' => '400', 'quantity' => 1), array('name' => 'Телефон', 'price' => '300', 'quantity' => 3), array('name' => 'Кроссовки', 'price' => '150', 'quantity' => 2),

);

Возвращает массив, который содержит:

* + Общую сумму покупок
  + Общее количество выбранных товаров

**Задача 2: Квадратное уравнение**

Написать функцию, которая решает квадратное уравнение. Функция принимает 3 аргумента (коэффициенты).

Возвращает:

* + - Массив с двумя корнями х1, х2, если D > 0
    - Один корень х, если D = 0
    - false, если D < 0

*Подсказки:*

Частично используйте код из прошлой практической работы.

**Задача 3: Удаление отрицательных элементов из массива (вариант 1)**

Есть массив с элементами (отрицательными и положительными). Нужно написать такую функцию*deleteNegtives(),* которая будет принимать массив, удалять из него элементы меньше 0 ивозвращать этот массив.

*Подсказки:*

Можно использовать цикл foreach для обхода элементов массива.

*Пример:*

// Сейчас $digits содержит отрицательные и положительные числа

$digits = array(2,­10, ­2, 4, 5, 1, 6, 200, 1.6, 95);

$digits = deleteNegtives($digits);

// Теперь $digits содержит только положительные числа

**Задача 4: Удаление отрицательных элементов из массива (вариант 2)**

Решить задачу №3 используя передачу аргумента по ссылке.

*Пример:*

$digits = array(2,­10, ­2, 4, 5, 1, 6, 200, 1.6, 95);

// Сейчас $digits содержит отрицательные и положительные числа deleteNegtives($digits);

// Теперь $digits содержит только положительные числа

*Все задачи необходимо сделать в отдельных файлах (например: task1.php) Оформить отчет с листингом кода*

**Практическая работа №4**

**«HTTP. Формы»**

**Задачи обучающегося:**

1. Научиться создавать страницы с формами для ввода данных

**Опорные понятия:**пользовательские формы.

**Планируемый результат:**

Студент должен

Уметь создавать формы различных типов

**Необходимое оборудование:**ПК, IDENetBeans, OpenServer

**Алгоритм деятельности обучающегося: Задание 1: Request&Response**

Посетить свои любимые сайты, посмотреть на запросы браузера (обратить внимание наиспользование методов GET и POST, отправляемые заголовки) и ответы сервера (статус ответа,заголовки, тело сообщения).

*Подсказка:* Используйте инструменты для разработчиков Вашего браузера.

**Задача 1: Использование форм**

Создать страницу с формой на 7 полей для текстового ввода (input, typetext) и кнопкой отправки(input, typesubmit).

Данные, отправленные формой приводятся к типу integer: используем функцию intval().

Далее из 7­ми введенных в форму значений вычисляются:

* максимальное значение
* минимальное значение
* среднее арифметическое значение.

Результаты выводятся после отправки формы на этой же странице.

*Подсказки:*

Данные можно записать в массив:

$a = intval($\_POST['a']);

...

$arr = array($a, $b, ...);

И далее применить функции max(), min() для нахождения максимального и минимальногозначений.

**Задача 2: Использование форм**

Создать страницу с формой, состоящей из:

* текстового поля (имя)
* поля выбора пола (как в примере в видео ­ inputtyperadio).

В зависимости от отправленных данных (значения поля "пол" ­ М или Ж) выводить приветствие.

*Пример:*

"Добро пожаловать, мистер Смит!" или

"Добро пожаловать, миссис Смит!"

*Все задачи необходимо сделать в отдельных файлах (например: task1.php) Оформить отчет с листингом кода*

**Практическая работа №5**

**«Cookie. Сессии»**

**Задачи обучающегося:**

1. Научиться создавать страницы с формами для ввода данных

**Опорные понятия:**пользовательские формы.

**Планируемый результат:**

Студент должен

Уметь создавать формы различных типов

**Необходимое оборудование:**ПК, IDENetBeans, OpenServer

**Алгоритм деятельности обучающегося:**

Задача 1: Время жизни cookie

Пользователь приходит на сайт. Используя cookie сделать так, что б впервые пришедшийпользователь видел фразу:

"Добро пожаловать, новичок!"

Если пользователь уже посещал сайт в течении последних 10­ти часов, выводить фразу: "С возвращением, дружище!"

*Подсказка:* Используйте инструменты для разработчиков Вашего браузера для просмотра иочистки текущих значений cookie.

**Задача 2: Дата и время последнего посещения**

Используя cookie реализовать вывод на страницу сообщения с датой и временем последнеговизита.

**Задача 3: Счетчик посещений**

Используя cookie реализовать вывод на страницу сообщения с количеством посещенийстраницы.

**Задача 4: Многостраничный тест**

Представим себе сайт с прохождением теста: на каждой странице находится вопрос, варианты ответа и кнопка “Далее”. Тест содержит некоторое количество страниц (допустим 3). На последней, 4­й странице пользователь должен получить результат теста.

Задача – создать подобный тест.

*Подсказки:*

* + Для создания страниц вопросов используйте формы (с элементамиtype=”radio”), которыебудут отправлять результаты на следующуюстраницу с вопросом или результатами.
  + Ответы пользователя на вопросы на предыдущих страницах можно (и нужно)сохранять в сессии
  + Для расчета результатов теста сравните ответы пользователя (хранятся всессии) с заранее определенными правильными ответами (могут храниться,например, в массиве или переменных).
  + После окончания текста сессию следует очистить

*Все задачи необходимо сделать в отдельных файлах (например: task1.php) Оформить отчет с листингом кода*

**«Базы данных»**

**Задачи обучающегося:**

1. Научиться создавать базы данных, создавать, удалять, изменять записи

**Опорные понятия:**пользовательские формы.

**Планируемый результат:**

Студент должен

Уметь создавать создавать базы данных, создавать, удалять, изменять записи

**Необходимое оборудование:**ПК, IDENetBeans, OpenServer

**Алгоритм деятельности обучающегося:**

Создайте базу данных со структурой 4 таблицы с заданными полями. Обратите внимание на типы полей, добавьте первичный ключ в каждую таблицу и AutoIncrement для первичногоключа.

Добавить в каждую таблицу 5­10 записей с правдоподобными данными:

* Несколько категорий новостей: спорт, политика, …
* Пользователи: имена, фамилии, логины, пароли, ... (последний визит не заполняйте)

Для новостей скопируйте тексты с новостных сайтов, укажите даты (с интервалом в несколько дней), поле preview оставить пустым, значение в category\_id соответствует значению id из таблицы news\_category, аналогично author\_id соответствует user.id

Комментарии: текст, дата, другие поля аналогично.

*Подсказка: используйте phpMyAdmin*

**Задача 1: INSERT**

Добавить 3 записи в таблицу user с любыми данными одним запросом INSERT.

**Задача 2: DELETE; AND и OR в условиях WHERE**

1. Написать запрос для удаления пользователей (таблица user) с именем "Томас".
2. Написать запрос для удаления пользователей с именем "Томас" и фамилией "Смит".
3. Написать запрос для удаления пользователей с именем "Томас" или именем "Джон".

*Подсказка:*(на английском языке. Если сложно читать ­ просто посмотреть примеры [запросов)http://www.techonthenet.com/mysql/and\_or.php](http://www.techonthenet.com/mysql/and_or.php)

**Задача 3: UPDATE**

1. Изменить имя (на "Тимофей") и фамилию (на "Опель") пользователю с идентификатором "3".
2. Изменить фамилию троим пользователям с наибольшими значениями id на "Смит".

*Подсказка:* используйте ORDER BY, LIMIT.

**Задача 4: SELECT**

1. Выбрать 3 последних новости из категории с идентификатором 2.
2. Выбрать всех пользователей с именем "Владислав" или "Елена".
3. В категории новости подсчитать количество новостей со статусом "1" и "0".

*Подсказка:* используйте GROUP BY, COUNT().

**Задача 5: SELECT + PHP**

Вывести список всех новостей со статусом "1" в виде списка:

<h1>Заголовок</h1>

<p>Короткое описание</p>

<ahref="/news/view/ИДЕНТИФИКАТОР">Читать далее</a>

*Подсказка:* используйте запрос с оператором SELECT и цикл foreach

**Практическая работа №6**

**«Объектно-ориентированное программирование»**

**Задачи обучающегося:**

* 1. Научиться создавать базы данных, создавать, удалять, изменять записи

**Опорные понятия:**классы и объекты.

**Планируемый результат:**

Студент должен

Уметь создавать функции используя принципы ООП

**Необходимое оборудование:**ПК, IDENetBeans, OpenServer

Используйте класс Car из занятия.

1. Добавить классу 3 свойства на свое усмотрение (например: количество дверей, стоимость,

...).Одному из свойств присвоить значение по умолчанию.

1. Создать 4 объекта класса Car.
2. Для двух объектов задать значения свойств, используя обращение к свойству (например,

$car1­>price).

1. Для двух оставшихся задать свойства используя конструктор (написать констуктор дляинициализации объекта).
2. Написать функцию fuelСonsumption() для расчета количества топлива, затраченного напоездку на заданную дистанцию. Используйте свойство объекта $fuel (расход топлива на 100 км).
3. Добавить в класс 3 числовых константы (со значениями, например, 2,5,12). Распечатать значенияконстант в коде программы (вне класса).
4. Добавить статический метод getMaxConstant(). Этот метод находит наибольшую из константкласса и возвращает её значение. Не забывайте, что статические методы принадлежат классу, ивызываются из контекста класса.

**Задачи обучающегося:**

* 1. Научиться создавать базы данных, создавать, удалять, изменять записи

**Опорные понятия:**классы и объекты.

**Планируемый результат:**

Студент должен

Уметь создавать функции используя принципы ООП

**Необходимое оборудование:**ПК, IDENetBeans, OpenServer

1. Создать класс Figure ­ плоская геометрическая фигура. У этого класса есть такие свойства:площадь, цвет. И константа: количество сторон.
2. Для класса Figure написать метод infoAbout(). Этот метод возвращает сообщение: "Это геометрическая фигура!".
3. От класса Figure унаследовать классы: Rectangle, Triangle, Square.
4. Добавить для Rectangle приватные свойства a,b ­ длины сторон.
5. Добавить для Square приватное свойство a ­ длина стороны.
6. Добавить для Triangle приватные свойства a, b, c ­ длины сторон.
7. Для каждого из классов Rectangle, Triangle, Square определить значение константы: количество сторон. Например, для квадрата: const SIDES\_COUNT = 4;
8. Создать конструкторы для классов Rectangle, Triangle, Square для инициализации значений длин сторон.
9. Для каждого из классов Rectangle, Triangle, Square реализовать метод getArea() ­подсчет площади. Методы возвращают значение площади. Подсказки ниже.
10. Для каждого из классов Rectangle, Triangle, Square переопределить метод infoAbout() так, что б он возвращал строку такого содержания: (пример для квадрата):

"Это класс квадрата. У него 4 стороны". Аналогично для других классов.

*Подсказка:* использовать константу класса для вывода количество сторон.

1. Сделать методы infoAbout() финальными.
2. Для каждого класса Rectangle, Triangle, Square создать по 2 объекта (с передачейзначений длин сторон в конструктор).
3. Вызвать для всех объектов методы getArea(), вывести результаты. Подсказки по математике:
4. Площадь прямоугольника:

S = a \* b;

1. Площадь квадрата:

S = a \* a;

1. Площадь треугольника: [http://www­formula.ru/index.php/2011­09­19­02­39­24/2011­09­19­03­26­12](http://www-formula.ru/index.php/2011-09-19-02-39-24/2011-09-19-03-26-12)

**Задачи обучающегося:**

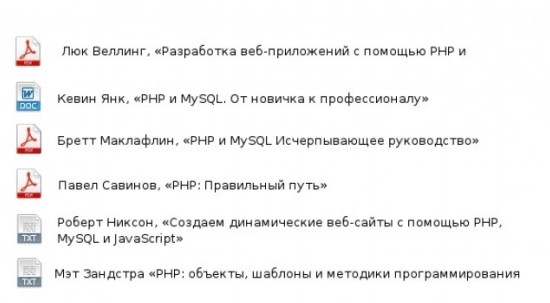
* 1. Научиться создавать базы данных, создавать, удалять, изменять записи

**Опорные понятия:**классы и объекты. **Планируемый результат:** Студент должен

Уметь создавать функции используя принципы ООП

**Необходимое оборудование:**ПК, IDENetBeans, OpenServer

В таблице books хранятся записи о книгах. Задача ­ получить эти записи и вывести так:



Нужно получить примерно такой html код:

<p>

<imgsrc="/images/BookPdf.jpg" alt="" />

<a href="/files/webappsdev.pdf">ЛюкВеллинг,Разработкавебприложений</a>

</p>

<p>

<imgsrc="/images/BookDoc.jpg" alt="" />

<a href="/files/phpmysql.doc">КевинЯнк, PHP MySQL</a>

</p>

1. Решить задачу нужно, используя полиморфизм.
2. Виды книг отличаются иконками.
3. Составляйте план решения задачи:
   * определите логические части программы
   * создайте файловую структуру
   * реализуйте необходимые классы, методы Структура таблицы book:
   * id
   * name
   * author
   * file\_path ­ путь к файлу на диске в папке /files/.
   * type ­ типзаписи: BookTxt, BookPdf, BookDoc.
   * sort\_order ­ порядковый номер книги в списке, учитывается при выборке записей из базыданных (ORDER BY)
4. Создать структуру файлов: books.php, classes.php, data.php.
5. Реализовать класс Book ­ базовый класс книги.

От класса Book унаследовать классы: BookPdf, BookTxt, BookDoc.

Для классов BookPdf, BookTxt, BookDoc реализовать метод printInfo() – печатаетнеобходимый блок для каждой книги.

1. Осуществить вывод списка книг подобно выводу публикаций в уроке.
2. Файлы хранить в папке /files/ сайта. При выводе книг подставлять значение из поля file\_path ваттрибутhref="..." ссылок. Тем самым, по нажатию на книгу мы сможем её скачивать.
3. Иконки сохранить в папке /images/ (найти любые иконки маленького размера из интернета). Привыводе книг подставлять в путь картинки значение из поля type.

**Информационное обеспечение реализации программы**

**Основные печатные издания**

# Усатая Т.В.и др. Дизайн-проектирование.- М.: ОИЦ «Академия», 2019. ББК 30.18:5-05я723

# Л. М. Корпан, А. А. Балканский, Л. П. Сопроненко [и др.] Проектная графика.- М.: ОИЦ «Академия», 2018.