МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ ГОРОДСКОГО ХОЗЯЙСТВА имени А. Н. БЕКЕТОВА

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения расчетно-графической работы

Основы JavaScript

по курсу

ИНФОРМАТИКА

(для студентов 1-го курса дневной формы обучения образовательно-квалификационного уровня бакалавр, направлений подготовки 6.030504 – Экономика предприятия и 6.030509 – Учет и аудит)

Харьков
ХНУГХ им. А. Н. Бекетова
2016

Методические указания для выполнения расчетно-графической работы «Основы JavaScript» по курсу «Информатика» (для студентов 1-го курса дневной формы обучения образовательно-квалификационного уровня бакалавр, направлений подготовки 6.030504 – Экономика предприятия и 6.030509 – Учет и аудит) / Харьков нац. ун-т гор. хоз-ва им. А. Н. Бекетова; сост. Б. И. Погребняк. – Харьков : ХНУГХ им. А. Н. Бекетова, 2016. – 24 с.

Составитель Б. И. Погребняк

Рецензент канд. физ.-мат. наук, доц. А. Б. Костенко

Рекомендовано кафедрой прикладной математики и информационных технологий, протокол № 1 от 31 августа 2015 г.

Оглавление

Стр.

Введение

Предпосылкой появления языка программирования *JavaScript* стала проблема, которая возникла в середине 90-х годов прошлого века, и которая была связана с необходимостью сделать Web-странички «живыми». То есть, разработчикам HTML-документов необходим был инструмент, который бы позволял динамически управлять всеми объектами Web-страниц, а также реагирующими на действия пользователя. И такой инструмент был создан Бренданом Эйком (Brendan Eich) из Netscape Communications. В разработке языка так же принимали участие сооснователь Netscape Communications Марк Андрессен и сооснователь Sun Microsystems Билл Джой.

Впервые новый язык был встроен в браузер Netscape Navigator 2.0 в 1995 году. В дальнейшем к развитию этого языка подключилась и Microsoft. Так, в 1996 году она выпустила свой вариант языка JavaScript, который получил название *JScript*. А первым браузером, который его поддерживал, был Internet Explorer версии 3.0. Но у фирмы Microsoft возникли определенные трудности использования этого языка связанные с его надежностью. Это привело к тому, что Microsoft в свои браузеры, помимо JavaScript, стал встраивать еще и интерпретатор языка *VBScript* (Visual Basic Script), который является ее авторским решением. Сейчас интерпретатор языка программирования JavaScript встраивается во все основные браузеры, именно поэтому они и могут выполнять скрипты на Web-страницах.

Сегодня JavaScript наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности Web-страницам. Однако, область его применения только этим не ограничивается. Он также используется как встраиваемый язык для программного доступа к объектам различных приложений. То есть, JavaScript – это полноценный язык, программы на котором можно запускать не только в браузере, но и на сервере (например, Node.js). Еще он широко используется для написания макросов в таких приложениях от Adobe, как Adobe Photoshop, Adobe Dreamweaver, Adobe Illustrator, а также входящих в состав пакета OpenOffice.org.

Основные теоретические сведения

Программа на языке JavaScript состоит из:

- *инструкций* и
- *комментариев*.

В свою очередь, инструкции тоже бывают двух видов:

1. инструкции, которые описывают элементы внутренних данных, требующихся в ходе выполнения программы – *инструкции-декларации* и
2. инструкции, которые описывают действия над этими элементами данных – *инструкции-команды*.

Элементы внутренних данных, которые используются в программах, бывают двух видов. Это:

- *переменные* и
- *константы*.

Всякая переменная характеризуется тремя параметрами:

- *именем*,
- *типом* и
- *значением*.

Имя переменной выбирается программистом произвольно, но, по-возможности, должно максимально точно соответствовать ее «смыслу». Оно может содержать любое количество латинских символов и цифр – ***без символа пробела и начинаться с «буквы»***; ***имя переменной, так же, не должно совпадать с ключевыми словами языка (например, таким как*** `var`***)***. Если имя переменной состоит из нескольких слов, то применяется так называемая «верблюжьяНотация» – когда каждое слово (кроме первого) начинается с большой буквы. Имя переменной, как правило, – существительное.

Тип переменной определяет множество значений, которые эта переменная может принимать, и операции, которые над ней могут выполняться. В языке программирования JavaScript определены следующие три основные (элементарных) типа данных:

- *логический* (`boolean`), который содержат всего два значения: `true` и `false`. Над ними можно выполнять такие операции, как:

  `&&` *логическое И* – возвращает `true`, если оба операнда имеют значение `true`.

  `||` *логическое ИЛИ* – возвращает `true`, если хотя бы один операнд имеют значение `true`.

  `!` *логическое НЕ* – унарная операция – возвращает `true`, если ее операнд имеют значение `false`.

- *строковой* (`string`) – самые распространенные на Web-страницах значения. К нему по умолчанию преобразуются другие типы данных в разнотипных операциях (если, конечно, нет явных указаний о преобразовании). Над строковыми данными можно выполнять единственную операцию – «+» – конкатенацию («склеивание», объединение) строк. В программе на JavaScript строковые значения заключаются в одинарные либо двойные кавычки (что бы интерпретатор мог отличить их от имен переменных и ключевых слов языка). При этом ***не допускается вложенность одинаковых кавычек!***

- *числовой* (`number`), который включают как целые, так и дробные значения. Над ними можно выполнять следующие операции:

`+` сложение,

`-` вычитание,

`*` умножение,

`/` деление,

`%` деление по модулю (остаток от деления нацело).

Операция деление по модулю `%` выполняется только над целочисленными значениями. Например, `7 % 2 = 1`.

Значения переменных по ходу выполнения программы могут меняться – потому они так и называются – переменные. Тип переменной так же может меняться, а вот имя – нет. Поэтому язык программирования JavaScript является *слабо типизированным* языком или языком *с динамической типизации*. Это значит, что одна и та же переменная по ходу выполнения программы, в различные моменты времени, может содержать значения различных типов. Или, более формально, тип переменной становится известным к моменту ее использования. То есть, интерпретатор JavaScript всегда знает, какого типа значение содержит данная переменная, а значит – какие операции к ней применимы. Есть *строго типизированные* языки программирования или языки *со статической типизацией* (например, Си), в которых тип переменной указывается в момент ее создания – в инструкции-декларации. Есть языки программирования, которые одновременно поддерживают и статическую и динамическую типизацию, например, Java. ***Язык программирования JavaScript поддерживает только динамическую типизацию!***

В языке программирования JavaScript переменные можно создавать двумя способами:

- *явно* – при помощи инструкции-декларации `var` и
- *неявно* – фактом своего первого появления в программе.

Второй способ хоть и кажется более простым, уступает первому в смысле потенциального появления трудно обнаруживаемых ошибок (например, непреднамеренное использование переменной, которая уже задействована где-то ранее в другом контексте). Поэтому в последних версиях языка введена специальная *инструкция-директива* `"use strict";`, которая, помимо других «строгостей», «заставляет» интерпретатор выдавать сообщение об ошибке в случае использования необъявленной переменной.

Явно переменные можно создавать в любом месте программы, даже после их использования (поскольку, интерпретатор «просматривает» код как минимум два раза), но лучше – в начале. После создания (объявления) переменной при помощи инструкции `var` она принимает специальное значение (тип) `undefined`. В этой же инструкции `var` можно так же присвоить переменной некоторое *начальное значение* (*инициализировать*) – тогда она сразу будет иметь не только имя, но и свой тип и значение. При этом распределение переменных по декларациям является произвольным, т. е. можно сосредоточить все переменные в одной декларации, но можно и, наоборот – для каждой переменной своя декларация. Руководствоваться всегда нужно здравым смыслом, и читабельностью программы, если нет на то других соображений.

Объекты внутренних данных программ, которые по ходу выполнения программы своих значений не меняют, (т. е., могут использоваться только справа в операции присваивания «=») называются *константами*. Они, так же как и переменные, характеризуются именем, типом и значением. Но, в отличие от последних, своих параметров в ходе выполнения программы не изменяют. Поэтому они и так называются константами (постоянными, неизменяемыми) Параметр имя для констант является необязательным. Вследствие этого, они бывают:

- *неименованные* и
- *именованные*.

Далее приведены примеры неименованных констант:

- `2` и `3.5` – целая и дробная числовые константы (для отделения целой и дробной части числа в программировании используется точка, а не запятая, как принято в математике);
- `"Это пример строковой константы"` и `'Это тоже'` – они заключаются в одинарные или двойные кавычки;
- `true` и `false` – логические константы имеют только два значения.

В языке программирования JavaScript нет специальных средств для работы с именованными константами. Однако, по общепринятой практике, для того, что бы визуально отличить именованную константу от переменной, их имена записываются только `ПРОПИСНЫМИ_СИМВОЛАМИ`, а отдельные слова в имени разделяются символом подчеркивания «_». В программировании именованные константы, в основном, используются в двух случаях:

- если одно и тоже значение в программе используется несколько раз, то имеет смысл создать именованную константу – чтобы в разных местах по-ошибке не присвоить различные значения, и
- для того, чтобы по программе небыли разбросаны различные «магические числа», значения которых не всегда очевидны из контекста. Например, если в начале программы создать именованную константу `WIDTH_WINDOW = 1024;`, то где-то в другом месте программы выражение `a = WIDTH_WINDOW;`, будет более наглядным и понятным, нежели просто `a = 1024;`, – переменной `a` присваивается именно ширина окна, а не какое-то непонятное число `1024`.

Присваивание – одно из самых фундаментальных действий в программировании. Инструкция присваивания делится на две части символом «=», который является знаком операции присваивания. Та часть, которая находится справа от него, определяет *значение*, а часть, расположенная слева, определяет *имя переменной*, которой должно быть присвоено данное значение. ***Операция присваивания всегда выполняется справа налево***, так что выражение `20 = x` вызовет ошибку, поскольку будет предпринята попытка присвоить `20` новое значение. Такое действие невозможно, поскольку `20` не является переменной, а есть целая числовая *неименованная константа* , значение которой не может быть изменено.

Все операции над данными, которые имеются во всех языках программирования, и в JavaScript в том числе, бывают трех видов:

- *унарные*, которые применяются к одному операнду (например, `-a` – меняет знак переменной `a` на противоположный),
- *бинарные*, которые применяются к двум операндам (например, `a + b` – сложение `a` и `b`) и
- *тернарные*, которые применяются к трем операндам. В JavaScript имеется единственный тернарный оператор – `? :`.

Действия, которые программа (скрипт) выполняет над исходными данными, образуют так называемый *вычислительный процесс* (*алгоритм*). Он описывается последовательностью из инструкций-команд. Выполняя такие инструкции-команды одну за другой, виртуальная JavaScript-машина производит необходимые действия по обработке информации.

Любая программа, на любом языке программирования, любой степени сложности, состоит из различных сочетаний следующих 3-х базовых алгоритмов:

1. *линейного*,
2. *разветвляющегося* и
3. *циклического*.

Линейный алгоритм является простейшим вычислительным процессом. В нем все действия по обработке информации выполняются от начала и до конца программы строго последовательно. Такой порядок выполнения действий называется естественным. То есть, для реализации линейного алгоритма ни каких специальных дополнительных управляющих конструкций в программу вводить не нужно.

В разветвляющемся алгоритме, в зависимости от некоторого условия, выполняется либо одна часть кода, а другая пропускается, либо – наоборот, пропускается первая, а выполняется – вторая. При помощи разветвляющегося алгоритма можно пропустить и, вообще, не выполнять некоторую часть программы. Для реализации разветвляющегося алгоритма в программу необходимо дополнительно вводить специальные управляющие инструкции.

Циклический вычислительный процесс служит для многократного повторения некоторой группы инструкций, которые называются *телом цикла*. А управляют этим процессом специальные инструкции – *заголовок цикла*. То есть, цикл состоит из двух частей: заголовка и тела. Циклы в программировании бывают двух видов. Если заранее известно, сколько раз должно выполниться тело цикла, то такие циклы называются *определенными* или *счетными циклами*. Если же момент завершения определяется в самом цикле, то такой цикл называется *неопределенным*. Проверка условия выполнения цикла может осуществляться как до начала тела цикла, так и после него. В первом случае она называется *предусловием*, во втором – *постусловием*. Однократное выполнение тела цикла называется *итерацией*.

JavaScript – это объектно-ориентированный язык программирования, который наиболее часто используется для управления поведением открытой в браузере Web-страницы. Программы на этом языке называются *скриптами* или *сценариями*. Они выполняется браузером при помощи встроенного в него интерпретатора. Скрипты подключаются напрямую к HTML-документу и, как только Web-страница загружается – тут же выполняются.

Скрипты на Web-страницу можно внедрять несколькими различными способами. Наиболее распространенный из них – при помощи парного тега `<script></script>`. При этом они могут располагаться как в области `head`, так и в области `body`: сценарии всегда обрабатываются и, если необходимо, запускаются на выполнение в том месте HTML-кода Web-страницы, где они присутствует.

Тег `<script>` языка HTML имеет два необязательных атрибута:

1. `type=` – в котором стандарт HTML-4 требует обязательного указания того, что скрипт написан именно на языке JavaScript, т. е. `type="text/javascript"`; в новом стандарте HTML-5 этот атрибут не

обязателен – по умолчанию предполагается что сценарий написан на языке JavaScript; и

2. `src=` – который содержит ссылку на файл с внешним сценарием.

Работа с внешними сценариями имеет некоторые особенности, которые заключаются в следующем:

- по умолчанию внешние файлы сценариев имеет расширение имени файла `.js`;
- внешний файл сценариев тегов языка HTML содержать не может (например, `<script></script>`) – в нем записываются только команды JavaScript;
- если подключен внешний файл сценариев, внутреннее содержание тегов `<script></script>` игнорируется.

Каждая инструкция языка программирования JavaScript, как правило, располагается в отдельной строке кода и заканчивается символом «точка с запятой» (`;`). В этом смысле она напоминает предложение естественного языка – какую-то законченную мысль, в конце которой ставится точка. Отступы и пустые строки в программе необязательны (интерпретатором они не контролируются), но весьма желательны, т. к. они подчеркивают ее структуру и улучшают «читабельность».

В языке программирования JavaScript (в отличие от языка HTML, который не чувствителен к регистру символов) различаются прописные и строчные буквы. Это значит, что имена переменных, функций и другие конструкции языка необходимо записывать так, как приводится в руководствах. ***Все ключевые (служебные, зарезервированные) слова языка программирования JavaScript записываются строчными буквами!***

В языке программирования JavaScript имеется несколько возможностей вывода информации из программы. Одна из них – при помощи функции `alert()`. Функция `alert()` выводит окно сообщения и приостанавливает выполнение скрипта, пока пользователь не нажмет **OK**. В качестве параметра она принимает заключенную в кавычки строку символов, которую необходимо

отобразить. Если в этой строке встречаются кавычки, то внешние кавычки должны отличаться от внутренних. То есть, если внутренние одинарные, то внешние должны быть двойными и, наоборот – если внутренние двойные, то внешние – одинарные (***вложенность одинаковых кавычек в JavaScript не допускается***). Если необходимо «разломить» выводимую строку на две, то в месте «разлома» ставится специальная комбинация символов «`\n`» – обратный слеш и строчная латинская буква `n`. Есди же выводимая строка состоит из нескольких частей, то для их объединения в одну используется оперция канкатенация – «+».

Если для вывода значений переменных в JavaScript используется функция `alert()`, то для ввода их значений – функция `prompt()`. Она имеет один обязательный параметр – текстовую строку-приглашение. Возвращает она введенные данные ***в виде значения строкового типа*** – если пользователь нажал кнопку **OK**, или `null` – если он нажал кнопку **Отмена**.

В любой компьютерной программе всегда присутствуют следующие три основных блока (части):

1. ввод исходных данных,
2. их обработка и
3. вывод результата.

Код программы на языке JavaScript, помимо инструкций, может содержать и *комментарии*. При интерпретации программы они, пропускаются и совершенно не влияют на ее выполнение. Комментарии адресованы тем, кто будут читать данную программу, и поясняют основные моменты ее реализации. Комментарии в языке программирования JavaScript бывают двух видов:

- *однострочные*, которые начинаются двумя наклонными черточками «`//`» и продолжаются до конца текущей строки;
- *многострочные и внутристрочные*, которые начинаются с символов «`/*`», а заканчивается – «`*/`», между которыми можно вставлять любой текст. Они могут быть записаны в любом месте программы, где

можно поставить пробел. Единственное ограничение в комментариях этого типа – они не должны быть вложенными, как, например:

```
/* Внешний /* Внутренний комментарий */ комментарий */
```

Поскольку код JavaScript может быть вставлен в Web-страницу, ***необходимо четко отличать комментарии в языке JavaScript от комментариев языка HTML*** (`<!-- Комментарий HTML -->`)!

Все ошибки, которые могут встретиться в программах, делятся на две категории:

- *синтаксически*, которые нарушают синтаксис языка и не позволяют программе вообще далее выполняться (например, пропущена парная кавычка, используется необъявленная переменная в «строгом» режиме и т. п) и
- *семантические* (времени выполнения, «прокол в сюжете») – они не блокируют выполнение программы, но она работает не правильно (например, некорректное применение типов данных); для поиска и устранения таких ошибок наиболее часто применяется трассировка – выполнение всего или части скрипта в пошаговом режиме с наблюдением динамики изменения переменных.

Для создания программ на JavaScript используется тот же инструментарий, что и для создания обыкновенных Web-страниц:

1. текстовый редактор и
2. браузер.

Причем, подойдут как самые простые, так и самые «навороченные».

Технология создания скриптов на JavaScript такая же, как и технология создания Web-страниц:

– в текстовом редакторе **1**) *отредактировать*, **2**) *сохранить*, **3**) *перейти в браузер*, **4**) *просмотреть* и **5**) *все с начала…* или

– при помощи традиционной последовательности сочетаний клавиш: **Ctrl+S** (*сохранить*) ⇨ **Alt+Tab** (*перейти в браузер*) ⇨ **F5** (*просмотреть*) ⇨ **Alt+Tab** (*вернуться в редактор – и все с начала…*).

Пример

Написать программу, которая вводит два числа, а выводит – их сумму. Ввод исходных данных и вывод результата выполнить с помощью окон диалога. В результирующем окне диалога программа должна вывести:

а) Ваши Фамилия Имя, а с новой строки – курс, группа и
б) с новой строки – результат вычисления в форме:
   1) формулу,
   2) подставленные в нее значения переменных,
   3) ответ.

Решение поставленной задачи заключается в подготовке двух файла следующего содержания:

1. Web-страница (например, `index.html`) и

```
<html>
  <head>
    <title>Расчетно-графическая работа</title>
    <meta charset=utf-8>
    <script src=script1.js></script>
  </head>
  <body>
  </body>
</html>
```

2. скрипт (`script1.js`)

```
"use strict"; // Строго! Все под контролем!

//Объявление переменных
var a, b, c;

// Ввод исходных данных
```

```
a = +prompt("Введите a");
b = +prompt("Введите b");

// Обработка данных
c = a + b;

// Вывод результата
alert("Иванова Ира\n1-й курс группа ОиА 2013-1\n" +
      "c = a + b = " + a + " + " + b + " = " + c);
```

Пояснения

Первый файл (`index.html`) – это обыкновенная Web-страница, в области `<head>` которой, при помощи тега `<script>` подключен внешний скрипт – файл `script1.js`. Второй файл (`script1.js`) – собственно программа (скрипт) на языке JavaScript инструкции которого выполняют такие действия:

- `"use strict";` – запрет неявного создания переменных;
- `var a, b, c;` – объявление (декларация, создание) переменных; `a` и `b` – исходные, `c` – результирующая;
- `a = +prompt("Введите a");` – ввод и преобразование к числовому типу переменной `a`;
- `b = +prompt("Введите b");` – ввод и преобразование к числовому типу переменной `b`;
- `c = a + b;` – вычисление результата;
- `alert("Иванова Ира\n1-й курс группа ОиА 2013-1\n" + "c = a + b = " + a + " + " + b + " = " + c);` – вывод результата; компоненты, которые заключены в кавычки (") – это текстовые константы, остальные – числовые значения, а объединяются они в одну

строку при помощи операции конкатенации – «+»; при объединении числовые значения преобразуются в строковые; пара символов «\n» служат для «разлома» строки – чтобы дальнейший вывод происходил с новой строки.

Выбор варианта Индивидуального задания

Вариант Индивидуального задания формируется в соответствии с:

- указанием преподавателя, или
- номером по списку группы, или
- по 2-м последним цифрам номера зачетной книжки.

По 2-м последним цифрам номера зачетной книжки вариант работы вычисляется как **остаток от целочисленного деления 2-х последних цифр номера зачетной книжки на 30** либо **последовательным вычитанием из 2-х последних цифр номера зачетной книжки по 30 до тех пор, пока остаток не будет меньше 30**. Например, две последние цифры номера зачетной книжки «07», тогда 7 % 30 = 7 т. е. номер варианта индивидуального задания будет равен «7»; если две последние цифры номера зачетной книжки будут, например, «68», тогда номер варианта будет: 68 % 30 = 8 или 68 – 30 = 38 – 30 = 8, т. е. «8».

Порядок оформления

Расчетно-графическая работа оформляется на листах формата А4 (210×297 мм) через полтора интервала. Текст может быть написан как от руки, так и распечатан на компьютере. Его объем не должен превышать 10 страниц. Все варианты заданий реализуются при помощи линейного вычислительного

процесса. Отчет по расчетно-графическому заданию должен быть представлен в соответствии со следующей структурой:

1. Титульный лист (пример оформления приведен в Дополнении).
2. Оглавление.
3. Введение.
4. Решение.
5. Заключение.
6. Список литературы.

На титульном листе указывается название работы, вариант, фамилия и группа студента. Во Введении необходимо показать основные особенности языка программирования JavaScript, область применения, его достоинства и недостатки. В разделе Решение приводится постановка задачи (условия), ее решения в виде кода Web-страницы и программы на языке JavaScript, описание программы и особенностей ее реализации. В Заключении необходимо сделать выводы по выполненной работе.

Индивидуальные задания

Для всех вариантов! *Ввод исходных данных и вывод результата выполнить с помощью окон диалога. В результирующем окне диалога программа должна вывести:*

а) Ваши Фамилия Имя, а с новой строки – курс, группа и

б) с новой строки – результат.

1. **Из градусов в радианы**. Написать программу перевода величины угла из градусов в радианы. 2π рад = 360^0.
2. **Из радианов в градусы**. Написать программу перевода величины угла из радианов в градусы. $360^0 = 2\pi$ рад.
3. **Из дюймов в метрическую систему**. Написать программу перевода длины отрезка, указанной в дюймах, в метрическую систему, т. е. выразить ее в метрах, сантиметрах и миллиметрах (1 дюйм = 2.54 см).
4. **Из метрической системы в дюймы**. Написать программу перевода длины отрезка, указанной в метрической системе, т. е. в метрах, сантиметрах и миллиметрах, в дюймы (1 дюйм = 2.54 см).
5. **Временной интервал**. Написать программу, которая вычисляет длину интервала времени, если его начало и окончание заданы в часах, минутах и секундах (в пределах одних суток). Результат вывести в тех же единицах измерения.
6. **Квадратное уравнение**. Написать программу, которая вычисляет корни квадратного уравнения с положительным дискриминантом, заданного своими коэффициентами. Определить так же погрешность вычисления, подставив полученные значения в исходное уравнение.
7. **Русские неметрические единицы длины**. Написать программу перевода длины отрезка, заданную в метрах, в русскую неметрическую систему

единиц длины, в которой 1 верста = 500 сажень, 1 сажень = 3 аршина, 1 аршин = 16 вершков, а 1 вершок = 44.45 мм.

8. **Вершина параболы**. Написать программу, которая вычисляет координаты вершины параболы $y = ax^2 + bx + c$.
9. **Приближение Sin x**. Функция y = Sin x на отрезке [0; π/2] хорошо аппроксимируется разложением: $y = x - x^3/6 + x^5/120$. Написать программу, которая для заданного значения аргумента x вычисляет значение функции y по этой формуле и сравнивает с точным значением, вычисленным с помощью стандартной функции Sin().
10. **Среднее арифметическое**. Написать программу, которая вычисляет среднее арифметическое чисел x, y и z.
11. **Площадь треугольника**. Написать программу, которая вычисляет площадь треугольника со сторонами a, b и c.
12. **Дробная часть числа**. Переменной d присвоить дробную часть положительного числа x.
13. **Третья цифра от конца**. Присвоить целой переменной i третью от конца цифру в записи целого положительного числа j. Например, если j = 657876, то i = 8.
14. **Первая цифра из дробной части**. Присвоить целой переменной i первую цифру из дробной части положительного числа с плавающей точкой x. Например, если x = 123.456, то i = 4.
15. **Сумма цифр трехзначного числа**. Целой переменной i присвоить сумму цифр целого трехзначного числа j. Например, если j = 567, то i = 18.
16. **Радиус**. Написать программу, которая вычисляет длину окружности, площадь круга, а также площадь и объем шара одного и того же заданного радиуса.
17. **Прямоугольный треугольник**. Написать программу, которая вычисляет периметр и площадь прямоугольного треугольника по длинам двух катетов.

18. **Произведение цифр четырехзначного числа**. Найти произведение цифр заданного четырехзначного числа.
19. **Обратный порядок цифр числа**. Определить число, полученное выписыванием в обратном порядке цифр заданного трехзначного числа.
20. **Часовая стрелка – угол**. Определить угол f (в градусах) между положением часовой стрелки в начале суток и ее положением в h часов, m минут и s секунд ($0 \le h \le 11$, $0 \le m \le 59$, $0 \le s \le 59$).
21. **Часовая стрелка – часы и минуты**. Определить полное количество часов h и полное количество минут m, прошедших от начала суток до того момента (в первой половине дня), когда часовая стрелка повернулась на f градусов (f – вещественное, $0 \le f < 360$).
22. **Из фунтов в килограммы и граммы**. Написать программу перевода веса, заданного в фунтах, в килограммы и граммы (1 фунт = 405.9 грамма).
23. **Из килограммов и граммов в фунты**. Написать программу перевода веса, заданного в килограммах и граммах, в фунты (1 фунт = 409.5 грамма).
24. **Покупка**. Написать программу, которая вычисляет стоимость покупки, состоящей из нескольких тетрадей, обложек к ним и карандашей.
25. **Поездка на дачу**. Написать программу, которая по заданному расстоянию до дачи, расходе бензина на 100 км. пробега автомобиля и стоимости 1 литра бензина определяет стоимость поездки туда и обратно.
26. **Денежный формат**. Написать программу, которая преобразует заданное дробное число в денежный формат. Например, 12.5 должно быть преобразовано к виду 12 грн. 50 коп.
27. **Площадь кольца**. Написать программу, которая определяет площадь кольца.
28. **Из гривны в доллары**. Написать программу, которая на основе текущего курса переводит сумму, указанную в гривнах, в доллары.
29. **Из долларов в гривны**. Написать программу, которая на основе текущего курса переводит сумму, указанную в долларах, в гривны.

30. **Объем текстового файла**. Написать программу, которая определяет объем файла в Кбайт, если рукопись состоит из указанного количества страниц текста. При этом на каждой странице расположено одинаковое количество строк, и каждая строка, в свою очередь, состоит из одинакового количества символов, а один символ соответствует 1 байту.

Список литературы

1. Гудман Д., Моррисон М. JavaScript. Библия пользователя, 5-е издание.: Пер. с англ. – М. : ООО “И.Д. Вильямс“, 2006. – 1184 с. : ил.
2. Флэнаган Д. JavaScript. Подробное руководство. Пер. с англ. – Символ-Плюс, 2008. – 992 с., ил.
3. Ресиг Д. JavaScript. Профессиональные приемы программирования. – СПб.: Питер, 2009. – 720 с.: ил.
4. Ресиг Д., Бибо Б. Секреты JavaScript ниндзя: Пер. с англ. – М. : ООО “И.Д. Вильямс“, 2013. – 416 с. : ил.
5. Дунаев В. Самоучитель JavaScript, 2-е изд. – СПб.: Питер, 2005. – 395 с.: ил.
6. Современный учебник JavaScript [Електронный ресурс] – Режим доступа: http://learn.javascript.ru/.

Дополнение

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ ГОРОДСКОГО ХОЗЯЙСТВА имени А. Н. БЕКЕТОВА

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА ПО ИНФОРМАТИКЕ

Вариант № 1

Выполнила
студентка
группы ОиА 2016-1
Иванова И.И.

Проверил
доц. каф ПМиИТ
Погребняк Б.И.

ХАРЬКОВ – ХНУГХ им. А. Н. Бекетова – 2016

Навчальне видання

Методичні вказівки

для виконання розрахунково-графічної роботи

«Основи JavaScript»

з курсу

«ІНФОРМАТИКА»

(для студентів 1-го курсу денної та заочної форм навчання напрямів підготовки 6.030504 – Економіка підприємств і 6.030509 – Облік та аудит)

(рос. мовою)

Укладач **ПОГРЕБНЯК** Борис Іванович

Відповідальний за випуск *О. Б. Костенко*

За авторскою редакцією

Комп'ютерне верстання: *Б. І. Погребняк*

План 2015, поз. 413М

Підп. до друку 25.05.2016 р.	Формат 60x84 1/16
Друк на ризографі	Умо. друк. арк. 1
Тираж 50 прим.	Зам. №

Видавець і виготовлювач:
Харківський національний університет міського господарства
імені О. М. Бекетова
вул. Революції, 12, Харків, 61002
Електронна адреса: rectorat@kname.edu.ua
Свідоцтво суб'єкта видавничої справи:
ДК № 4705 від 28.03.2014 р.