

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению лабораторных работ
по дисциплине «Web-технологии»
для студентов направления подготовки
09.03.01 Информатика и вычислительная техника

Ставрополь 2026

Содержание

Лабораторная работа 1. Язык разметки гипертекста HTML. Основные понятия

Лабораторная работа 2. Форматирование текста. Управление цветом

Лабораторная работа 3. Гиперссылки.

Лабораторная работа 4. Работа со списком.

Лабораторная работа 5. Вставка графических изображений

Лабораторная работа 6. Таблицы HTML-документов

Лабораторная работа 7. Формы в HTML.

Лабораторная работа 8. Технология CSS

Лабораторная работа 9. Внедрение объектов JavaScript в HTML-документ

Лабораторная работа 10. Использование базовых конструкций языка JavaScript

Лабораторная работа 11. Функции. Рекурсивные функции

Лабораторная работа 12 . Алгоритмы работы со строками

Лабораторная работа 13. Массивы с числовыми индексами

Лабораторная работа 14. Объектно-ориентированное программирование в JavaScript

Лабораторная работа 15. Внутренние объекты JavaScript. Потомки объектов JavaScript

Лабораторная работа 16. Объекты как ассоциативные массивы

Лабораторная работа 17. Работа со строками и регулярными выражениями

Лабораторная работа 1

Тема: Язык разметки гипертекста HTML. Основные понятия

Цель работы: ознакомить студентов с основными понятиями языка HTML, структурой HTML-документа, обязательными метками, комментариями, способами форматирования текста, физическими и логическими стилями, приобретение навыков создания простейших статических Web-документов.

Теоретическая часть

Основы языка разметки гипертекста - HTML

Базовым элементом языка разметки гипертекста является - ТЕГ (дескриптор, маркер). Тег всегда заключен между скобками < > и имеет следующий вид: <ТЕГ параметр1="ЗНАЧЕНИЕ" ... параметрN="ЗНАЧЕНИЕ">

Теги бывают одиночными и контейнерными. *Контейнером* называется пара: открывающий <ТЕГ> и закрывающий </ТЕГ>.

<ТЕГ> Содержимое контейнера </ТЕГ>

Открывающий тег служит для указания программе-браузеру начала какого-либо объекта или задания свойств объектов помещенных в контейнер. Закрывающий тег служит для указания программе-браузеру о конце объекта или окончания применения свойств, заданных в открывающем теге. Параметры (атрибуты) тега задают значения свойств данного объекта или объектов помещенных в контейнер. Значения свойств, содержащие пробелы, берутся в кавычки, в остальных случаях кавычки можно опустить.

HTML документ представляет собой обычный текстовый файл, содержащий маркированный тегами форматирования текст, а так же заданные специальными тегами ссылки на графические и прочие файлы мультимедиа, ссылки на другие документы HTML и ресурсы Internet.

Браузер (Browser) - программа для просмотра Web-страниц.

Структура HTML-документа

Документ HTML начинается открывающим тегом <HTML> и заканчивается за-крывающим тегом </HTML>. Между данной парой контейнерных тегов располагаются две другие основные части HTML документа: заголовок заключенный в контейнер <HEAD>...</HEAD> и тело документа в контейнере <BODY>...</BODY>. Таким образом, структура простого HTML документа выглядит примерно так:

Структура HTML-документа

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.2//EN" >
<HTML>
<HEAD> Заголовок документа </HEAD>
<BODY> Тело документа </BODY>
</HTML>
```

Объявление <!DOCTYPE>

Элемент <!DOCTYPE> должен первым указываться в документе HTML (теоретически). Он сообщает серверу WEB способ обработки документа и то, какие дескрипторы могут находиться на странице, хотя чаще всего он игнорируется браузерами. Поэтому его применение строго не обязательно.

Синтаксис: <!DOCTYPE HTML "текст""URL">

Здесь текст определяет версию HTML , а URL позволяет браузерам пользователей загрузить DTD например:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://WWW.W3.ORG/TR/REC-HTML40/strict.dtd">
```

Тэг <HTML>

Тэг <HTML> определяет границы документа HTML, ему соответствует конечный тэг </HTML>. Между этими двумя тэгами располагается собственно весь документ. Как и <!DOCTYPE> тэги <HTML> и </HTML> - не являются строго обязательными. Но, все-таки, их использование является правилами хорошего тона т.к. браузеры у пользователей могут быть всякие и не известно - насколько корректно они визуализируют такой код.

В дополнение к обязательной структуре настоятельно рекомендуется вставлять различные структурные детали.

Каждый HTML документ должен содержать основную информацию о его происхождении.

Если Вы стремитесь к тому, чтобы люди отыскиали Ваш документ по соответствующим связям, важность предоставления информации о его происхождении становится очевидной. Когда пользователь найдет Ваш документ с помощью, например, поискового ресурса AltaVista, он, вероятнее всего, захочет узнать, к какому виду относится документ. Поэтому каждый файл HTML должен предоставлять самую основную информацию (или связи к информации) о его происхождении и природе. Например, в собрании книгоподобных документов, разделенных на малые файлы, каждый файл должен содержать, по крайней мере, связь к "первой странице" "книги" (home page).

О происхождении документа должна быть представлена, по крайней мере, следующая информация:

Автор документа, имеющий уникальное имя. При этом должна быть задана связь с домашней страницей автора. Если у документа несколько авторов, определите их всех, а также роль каждого из них; например, ведущего автора, редактора, действующего спонсора, а также лиц, формально отвечающих за документ. Дата создания документа или его последней модификации, или и та и другая. Представляемая дата должна быть понятна во всем мире; в частности, название месяца лучше писать словом, а не цифрой. Контекст документа и его статус, например: часть официальной документации компании об одном из ее продуктов, или часть личной информации о хобби автора, или что-то другое. Адрес (URL) документа. Такая информация зачастую чрезмерна, однако она может быть очень полезной, когда кому-то нужна копия именно того документа, который

он нашел. Лучше не полагаться на браузер (и пользователя), добавляющих такую информацию, когда сделана копия документа.

Элемент <BODY>

Элемент <BODY> предназначается для выделения той части документа, которая будет визуализирована для пользователя. Он имеет как начальный, так и конечный теги. Начальный тег <BODY> может иметь несколько атрибутов.

Заголовочные тэги

Элемент <HEAD> </HEAD> определяет заголовок документа.

BASE - базовый, основной URL. Цель: задание базового URL для относительных URL в документе (например, в атрибутах HREF элемента A). Этот элемент часто используется для отображения документов. Например: <BASE href="http://foo.com/index.html">.

Элемент BASE непосредственно не отображается в документе. Основной синтаксис <BASE HREF="URL">. Так как в документе допускается только один элемент BASE, Вы не можете иметь различные базовые URL в различных частях файла HTML. При отсутствии элемента BASE в документе URL самого документа становится базовым в пределах документа.

META - метаинформация

Используется для задания метаинформации (информации о документе), т.е. пар имя/значение, описывающих свойства документа, например, авторство, истечение даты, список ключевых слов и т.д. Типичное отображение. Элементы META не влияют на отображение самого документа. Они могут давать некоторый эффект при представлении информации о документе, например, в верхнем окне браузера или в ответе на запрос от поискового средства.

Синтаксис:

<META NAME=имя элемента метаинформации CONTENT=содержимое информации>
или

<META HTTP-EQUIV=имя элемента метаинформации CONTENT=содержимое информации>.

Примеры:

<META NAME=DESCRIPTION CONTENT= "An extensive guide to writing HTML 3.2 documents, with ../../examples and practical advice.">

<META NAME=KEYWORDS CONTENT= "structural HTML, logical markup">.

Тег META влияет на индексирование документа, когда он включается в базу данных поискового сервера.

TITLE - "внешний" заголовок (титул)

Используется для задания обязательного "внешнего" заголовка документа.

Титул может выводиться в окне заголовка программы просмотра; в списке результатов поиска, возвращаемых поисковым сервером; в горячем списке, определяемом пользователем; списке истории и т.д.

Основной синтаксис <TITLE>последовательность символов</TITLE>

В TITLE можно использовать escape последовательности, например, < (для <) и ¨ (для д), но никакие теги HTML не разрешены, поэтому Вы не можете задавать в заголовке размеры шрифтов или выделения.

Примеры <TITLE>A study of population dynamics</TITLE> Примечания
Написать хорошее заглавие - очень важно, так как списки результатов поиска, возвращаемые поисковым сервером, могут использовать его.

Комментарии

Файл HTML может содержать комментарии, дающие пояснения для человека, читающего HTML код. Комментарии не влияют каким-либо образом на представление документа, т.е. они игнорируются бро-узером. Вы можете начать комментарии с четырехсимвольной последовательности <!-- (знак "меньше чем", восклицательный знак, два дефиса) и завершить его трехсимвольной последовательностью -> (два дефиса, знак "больше чем"). Например: <!-- Написано Иваном Ивановым ->.

Пример 1

```
<HTML>
<HEAD>
<TITLE> моя первая страница </title>
<META NAME="AUTHOR" CONTENT=" SIDOROV SIDOR ">
<META NAME="KEYWORDS" CONTENT=" Животные, природа, фауна ">
</head>
<BODY>
Здравствуйте, это моя первая страница.
<!--<PRE> КОММЕНТАРИЙ </PRE>-->
<BR>
Добро пожаловать! :)
</body>
</html>
```

Оборудование и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 7.

Задания

Указания к выполнению:

1. Создайте на диске отдельную директорию (папку) для будущей страницы.

2. Откройте текстовый редактор Блокнот (notepad) и наберите в нем структуру HTML-документа, которая приведена в примере 1.

3. Сохраните этот документ, присвоив ему имя *.html

4. Откройте с помощью браузера ваш документ (не закрывайте блокнот, он нам еще пригодится).

Если вы что-то изменили в вашем *.html документе (в Блокноте), то, не забудьте сохранить изменения, а чтобы посмотреть как это выглядит, надо нажать в браузере кнопку ОБНОВИТЬ.

Если изменений не видно, то это значит, что вы где-то что-то неправильно написали, или забыли сохранить документ.

5. Вернитесь к сохраненному в *Блокноте* файлу.

6. Внесите в него следующие изменения: пусть это будет ваша первая страничка. Укажите в ней ваши фамилию, имя, ФИО родителей, братьев и сестер, свои увлечения. Используйте для этого форматирование абзацев.

7. В строке `<title>` укажите: "*Домашняя страничка (ваше имя и фамилия)*".

8. Сохраните файл как *page2.htm*.

9. Просмотрите результат в браузере, при необходимости отредактируйте файл при помощи Блокнота.

Содержание отчета

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать: название дисциплины, название лабораторной работы, фамилию и инициалы студента, номер группы, фамилию и инициалы преподавателя. Основная часть лабораторной работы должна содержать:

- вариант вашего задания;
- распечатку страниц выполненного задания;
- выводы по проделанной работе.

Защита работы

Защита работы заключается:

- в выполнении варианта своего задания и оформления отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- демонстрации преподавателю выполненного задания;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Основы работы в Web-среде».

Контрольные вопросы

1. Охарактеризуйте язык тегов HTML.
2. Какова структура HTML-документов.
3. Перечислите обязательные метки и охарактеризуйте их.
4. Дайте определения следующих понятий: гипертекст, гиперссылка, Web-сайт, URL-адрес в Интернете, фрейм апплет скрипт, браузер элемент, тэг, атрибут.

Лабораторная работа 2

Тема: Форматирование текста. Управление цветом

Цель работы: научиться форматировать шрифт и текст на Web-странице, научиться создавать цветной фон Web-страницы, использовать шрифт различного цвета.

Теоретическая часть

Форматирование текста

Текст – единственный объект Web-страницы, который не требует специального определения. Иными словами, произвольные символы интерпретируются по умолчанию как текстовые данные. Для форматирования текста существует большое количество элементов. Форматировать текст можно с помощью традиционных элементов: выделять фрагменты курсивом, полужирным шрифтом, выбирать шрифт, размер и цвет шрифта, выравнивать текстовые фрагменты. Если автору не хватает простых вариантов форматирования текста, он может прибегнуть к таблицам стилей, которые существенно расширят возможности языка HTML по форматированию. Рассмотрим стандартные элементы языка HTML, позволяющие форматировать текст.

Форматирование абзаца

Элемент <P> Этот элемент задает один из способов разбиения текста на абзацы. Он может иметь вложенный атрибут align, который указывает отступ left, center, right, justify. Каждый следующий абзац игнорирует, заданное для предыдущего абзаца значение align.

Синтаксис:

<P align=отступ> Текст абзаца </P>

Элемент абзаца разделяет фрагменты текста вертикальным отступом. Он позволяет использовать только начальный тэг, так как следующий элемент </p> обозначает не только начало следующего абзаца, но и конец предыдущего. В тех случаях, когда по смыслу необходимо обозначить

завершение абзаца, можно использовать и конечный тэг. Вместе с элементом абзаца можно использовать и атрибут выравнивания **align**:

- `<p align="left">` – выравнивание по левому краю;
- `<p align="right">` – выравнивание по правому краю;
- `<p align="center">` – выравнивание по центру;
- `<p align="justify">` – выравнивание по ширине. Выравнивание по ширине – опасный параграф. Он не работает в старых версиях браузеров (т.е. текст выровняется в таком случае по левому краю документа).

Пример 1

```
<HTML>
<HEAD>
<TITLE> Использование различных типов выравнивания </title>
</head>
<BODY>
<P align="justify"> Элемент абзаца разделяет фрагменты текста вертикальным
отступом. Вместе с элементом абзаца можно использовать и атрибут выравнивания
align: </p>
<P align="left"> align="left" – выравнивание по левому краю; </p>
<P align="center"> align="center" – выравнивание по центру; </p>
<P align="right"> align="right" – выравнивание по правому краю; </p>
<P align="justify"> align="justify" – выравнивание по ширине.</p>
</body>
</html>
```

**
**

Элемент, обеспечивающий принудительный переход на новую строку. Он имеет только начальный тэг. В месте его размещения строка заканчивается, а оставшийся текст печатается с новой строки.

Пример 2

```
<HTML>
<HEAD>
<TITLE> Принудительный перенос строки </title>
</head>
<BODY>
Уронили мишку на пол,
<BR> Оторвали мишке лапу,
<BR> Все равно его не брошу
<BR> Потому, что он хороший.
</body>
</html>
```

Если таким образом расставить элемент `<br>` в этом стихотворении, то в экране браузера мы увидим стандартным образом написанное четверостишие на четырех строках.

`<NOBR>...</nobr>`

Этот элемент по своему действию является прямой противоположностью предыдущему. Текст, заключенный между его тегами, будет выведен в одну строку. Длинная строка не уместится на экране, и для ее просмотра придется использовать горизонтальную полосу прокрутки.

`<pre>...</pre>`

Элемент для обозначения текста, отформатированного заранее (*preformatted*). Подразумевается, что текст будет выведен в том виде, в каком он был подготовлен пользователем. Текст, заключенный между метками `<PRE>` и `</pre>` (от английского *preformatted* — предварительно отформатированный), выводится браузером на экран как есть — со всеми пробелами, символами табуляции и конца строки. Во всех других случаях браузер игнорирует эти символы. Возможен и обратный эффект: если пользователь введет текст как одну длинную строку, то она не будет разорвана браузером, а уйдет за границы окна программы. В этом смысле элемент *PRE* работает так же, как элемент *NOBR*. По умолчанию для отформатированного заранее текста выбирается моноширинный шрифт. Этот элемент удобно использовать для демонстрации листингов программ или для вывода текстовых документов, переформатирование которых может привести к искажению их смысла.

`<BLOCKQUOTE> ... </blockquote>`

Текст, заключенный между метками `<BLOCKQUOTE>` и `</blockquote>`, выводится браузером на экран с увеличенным левым полем (с отступом).

`<CENTER>...</center>`

Элемент для центрирования текста, а точнее, любого содержимого. Не является общеупотребительным. В тех случаях, когда это возможно, вместо него используют атрибут *align = "center"*.

<DIV>...</div>

Элемент, похожий на предыдущий. Он позволяет выравнивать содержимое по левому краю, по центру или по правому краю. Для этого стартовый тег должен содержать соответствующий атрибут:

- *align = "left"*
- *align = "center"*
- *align = "right"*

Форматирование шрифта HTML допускает два подхода к шрифтовому выделению фрагментов текста – логическое и физическое форматирование.

В группу тегов логического форматирования входят теги, отображающие на экране монитора элементы документа, таким образом, как установлено по умолчанию в спецификации языка разметки HTML. Переопределить их параметры или свойства нельзя, за исключением ситуаций использования стилевых шаблонов CSS и обособления тегами физического форматирования. Результат действия разных тегов логического форматирования визуально может совпадать, ибо основное их предназначение заключается в логическом выделении отдельных элементов HTML.

Теги физического форматирования позволяют разработчику визуально изменять вид текста, делая его жирным или наклонным, варьируя его параметры и значения. Под физическом стилем принято понимать прямое указание браузеру на модификацию текущего шрифта.

...

Выделение текста полужирным шрифтом (от слова *Bold*), - очень популярный элемент. Использование полужирного шрифта – прием,

позаимствованный из текстовых редакторов. Т.е. текст находящийся внутри тега будет жирным.

<I>...</i>

Выделение текста курсивом (от слова *italic*). Т.е. текст, находящийся внутри тега *под наклоном*

Оборудование и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 7.

Задания

Указания к выполнению задания 1

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.
2. В элементе *<title>* укажите название странички "*Задание 1. Форматирование текста на Web-странице по образцу*".
3. Отформатируйте следующий текст согласно указаниям таблицы.

Текст	Применить следующие форматирование
Автомобили	Заголовок Н1
Легковые	Заголовок Н2
Отечественные	выравнивание слева, полужирный шрифт, цвет шрифта зеленый
Импортные	выравнивания справа, полужирный шрифт, цвет шрифта оранжевый
Тестдрайв	выравнивание по центру, полужирный курсив
Салоны продаж	Выравнивание по центру, размер шрифта 3, цвет шрифта красный
Стихотворение	цвет шрифта синий, выравнивание по центру
Уронили мишку на пол, Оторвали мишке лапу. Все равно его не брошу - Потому что он хороший.	Выравнивание по левому краю, размер шрифта 4

Указания к выполнению задания 2

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.
2. В элементе `<title>` укажите название странички "Задание 2. Форматирование текста на Web-странице по образцу".
3. Напишите HTML-файл, чтобы при его выполнении получился текст следующего формата:

Элементы форматирования текста

Здесь находится цитата...

• Фрагменты текста можно выделять: **жирным** или *наклонным* шрифтом. Кроме того, можно включать в текст фрагменты с фиксированной шириной символа (имитация пишущей машинки).

- Текст может быть или подчеркнутым.
- Можно также включать элементы, создающие эффект нижнего индекса или верхнего индекса.

- Использование увеличения шрифта – BIG
- Использование уменьшения шрифта – SMALL
- Кроме физических существуют также логические стили
- Использование акцента – EM
- Использование сильного акцента – STRONG
- Фрагмент исходного текста CODE
- Использование образца – SAMP
- Текст, вводимый с клавиатуры – KBD
- Значение переменной – VAR
- Использование элемента CITE
- Так выглядит формат адреса (элемент ADDRESS)
- Использование элемента CENTER
- Использование элемента DIV
- Элемент BR обеспечивает разрыв строки

Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать: название дисциплины, название лабораторной работы, фамилию и инициалы студента, номер группы, фамилию и инициалы преподавателя. Основная часть лабораторной работы должна содержать:

- вариант вашего задания;
- распечатку страниц выполненного задания;
- выводы по проделанной работе.

Защита работы

Защита работы заключается:

- в выполнении варианта своего задания и оформления отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- демонстрации преподавателю выполненного задания;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Основы работы в Web-среде».

Контрольные вопросы

1. Дайте характеристику форматированию абзаца.
2. Дайте характеристику форматированию шрифта.
3. Логические стили в HTML - документах.
4. Комментарии в HTML-документах.
5. Escape последовательности.
6. Что такое цветовая гамма HTML-документа
7. Сочетание цветов в документе

Лабораторная работа 3

Тема: Гиперссылки.

Цель работы: познакомиться с основными принципами построения гипертекстовых ссылок.

Теоретическая часть

Гипертекстовые ссылки

HTML предлагает множество условных оборотов для текстовых и структурированных документов, но что отличает его от большинства других языков разметки - его возможности разметки гипертекста и интерактивных документов. Ссылки связывают один ресурс Web с другим. Несмотря на простоту, ссылки стали основным залогом успеха Web.

Гипертекст (Hyper Text) - это многомерный текст, т.е. такая организация документов, при которой один документ или текст может включать в себя разнонаправленные ссылки. Эти ссылки, называемые гипертекстовыми ссылками или гиперссылками (Hyper Text links, hyperlinks), позволяют просматривать документ в любом необходимом порядке. Объединение одномерных текстов с включенными в них гиперссылками называется гипертекстом.

Самым простым примером гипертекста является система контекстной помощи Microsoft Windows.

Любая гиперссылка состоит из двух частей: *указателя ссылки* ("якоря" - от англ. anchors) и *адреса ресурса*, которым может быть любым ресурсом Web (например, изображением, видеоклипком, звуковым файлом, программой, документом HTML, элементом в документе HTML и т.д.), на который необходимо осуществить переход.

Внешне отличить гиперссылку от обычного текста очень просто: при наведении курсора мыши на ссылку указатель принимает вид руки с указательным пальцем, как бы показывающим, что этот текст содержит гиперссылку. Сама ссылка подчеркивается (в случае если указателем является текст).

В качестве указателя может выступать текст (отдельное слово, фразы и даже целые страницы текста) и графические изображения. В ряде случаев возможно объединение графики и текста в рамках единого указателя ссылки.

Гипертекстовые ссылки являются ключевым компонентом, делающим WEB привлекательным для пользователей. Добавляя гипертекстовые ссылки (далее - ссылки), вы делаете набор документов связанным и структурированным, что позволяет пользователю получать необходимую ему информацию максимально быстро и удобно. Ссылки имеют стандартный формат, что позволяет браузеру интерпретировать их и выполнять необходимые функции (вызывать методы) в зависимости от типа ссылки.

Ссылки могут указывать на другой документ, специальное место данного документа или выполнять другие функции, например запрашивать файл по FTP-протоколу для отображения его браузером.

URL может указывать на специальное место по абсолютному пути доступа, или указывать на документ в текущем пути доступа, что часто используется при организации больших структурированных WEB-сайтов.

URL

HTML использует URL (Uniform Resource Locator) для представления гипертекстовых ссылок и ссылок на сетевые сервисы внутри HTML-документа. Первая часть URL (до двоеточия) описывает метод доступа или сетевой сервис. Другая часть URL (после двоеточия) интерпретируется в зависимости от метода доступа. Обычно, два прямых слэша после двоеточия обозначают имя машины:

Uniform Resource Locator имеет следующий формат:

method://servername:port/pathname#anchor.

Структура гиперссылки

Указатель ссылки описывается тегом **<A>**, а адрес перехода реализован с помощью параметра **HREF**, значением которого является путь к тому или иному интернет-ресурсу.

`<A>...</a>`

Если файл находится в том же каталоге, что и документ, на который сделана ссылка, то полный путь к документу указывать необязательно. Достаточно использовать сокращенные версии адресов, называемые относительными адресами.

Чаще всего используются следующие шаблоны:

- *текст* `<A href="Адрес ссылки"> текст для щелчка </a>`
- `<A href="Адрес ссылки"> <IMG src="Ссылка на рисунок"> </a>`

Первый шаблон применяется в том случае, когда гиперссылка встречается в тексте. Атрибут *href* может указывать на ресурс **Internet**, файл на локальном диске или метку внутри текущей страницы. Текст, расположенный внутри элемента `<A>`, представляет собой видимую часть гиперссылки. Именно на нем должен щелкнуть пользователь, чтобы осуществить переход. Браузер выделяет этот фрагмент цветом, а после использования гиперссылки меняет цвет, чтобы обеспечить подсказку.

link - определяет цвет выделенного элемента текста, при нажатии на который происходит переход по гипертекстовой ссылке.

– **link** = `"#FF0000"` - цвет гипертекстовой ссылки. Насыщенность красным - *FF* (255), зеленым и синим - *00* (ноль). Результат - красный цвет.

– **vlink** - определяет цвет ссылки на документ, который уже был просмотрен ранее.

– **alink** - определяет цвет ссылки в момент, когда на нее указывает курсор мыши и нажата ее правая кнопка, то есть непосредственно перед переходом по ссылке.

Пример 1

```
<HTML>
<HEAD>
<TITLE> Использование текстовой гиперссылки</TITLE>
</HEAD>
<BODY>
<Любая гиперссылка состоит из двух частей:
<A HREF ="1.html"> указателя ссылки </a>
```

("якоря" - от англ. anchors) и
 адреса ресурса,
которым может быть любым ресурсом Web, на который необходимо осуществить переход.
</BODY>
</HTML>

Оборудование и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 7.

Задания

Задание 1. Создание простейшей гиперссылки

Преобразуйте страницы в простой сайт, связав отдельные части с помощью гиперссылок. Необходимо использовать все виды гиперссылок.

Обязательные страницы:

- Об авторе.
- Мой ВУЗ - о СКФУ.
- Моя группа.
- Моя будущая профессия.
- Мои увлечения или хобби.

• На страничке *Моя будущая профессия* создайте в конце документа гиперссылку для перехода в начало этого же документа (якорь).

• Откройте в браузере страничку *Об авторе* и убедитесь, что теперь можно перейти с любой странички на любую другую из созданных.

Задание 2. Цвет гиперссылок

• На страничках, созданных в задании 1, в тэге <BODY> определите цвет гиперссылок:

- все гиперссылки на странице - белые;
- активные гиперссылки - красные;
- посещенные гиперссылки - серые.

• Откройте в браузере первую страничку и, переходя по ссылкам со странички на страничку, убедитесь, что цвет гиперссылок задан верно.

Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать: название дисциплины, название лабораторной работы, фамилию и инициалы студента, номер группы, фамилию и инициалы преподавателя. Основная часть лабораторной работы должна содержать:

- ☐ вариант вашего задания;
- ☐ распечатку страниц выполненного задания;
- ☐ выводы по проделанной работе.

Защита работы

Защита работы заключается:

- ☐ в выполнении варианта своего задания и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- ☐ демонстрации преподавателю выполненного задания;
- ☐ в ответах на контрольные вопросы по лабораторной работе;
- ☐ в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Основы работы в Web-среде».

Контрольные вопросы

1. Понятие гипертекста
2. Понятие гиперссылки.
3. Структура гиперссылки
4. Относительный адрес.
5. Абсолютный адрес.
6. Внешние и внутренние гиперссылки.

Лабораторная работа 4

Тема: Работа со списком.

Цель работы: научиться создавать списки на Web-странице, научиться размещать графические изображения на Web-странице.

Работа со списками

Очень важный элемент Web-страничек – *списки*. Использование списков в качестве средства представления информации объясняется следующим:

1. Информация в виде списка позволяет создавать большие массивы данных на отдельные четкие фрагменты, которые человеку гораздо удобнее воспринимать, чем весь поток целиком.
2. Списки позволяют создавать понятную и логичную внутреннюю структуру информационных данных ориентироваться в которой просто и удобно.
3. Использование списков удобно для отображения определенных пошаговых и прочих последовательных процессов. Список отличается от обычного текста тем, что пользователю не надо думать о нумерации его пунктов: эту задачу программа берет на себя. Если список дополняется новыми пунктами или укорачивается, нумерация корректируется автоматически. В случае нумерованных списков программа ставит перед каждым пунктом маркеры: кружки, прямоугольники, ромбы или другие изображения. В результате список принимает удобочитаемый вид.

Спецификация HTML предусматривает три основных типа списков:

- маркерованные списки;
- нумерованные списки;
- списки определений.

Маркированный (нумерованный - Unordered List) список.

Маркированный список представляет собой нумерованный или неупорядоченный перечень элементов, для заголовка которых используются специальные маркеры. В качестве маркеров выступают специальные символы, называемые буллетами (bullets). Текст, расположенный между тегами **** и ****, воспринимается как нумерованный список.

Элемент **, используется с целью задания нумерованных списков, имеет атрибут `type=disc,square, или disc` для задания вида маркера. Элемент ** включает в себя дополнительный элемент ****, который задает элементы списка.

Синтаксис:

<UL type=disc >

**** элемент списка

**** элемент списка ****

Для элемента *UL* возможными значениями атрибута *type* являются *disc*, *square* и *circle*, где

- **disc** – маркер в виде закрашенного круга
- **circle** – маркер в виде незакрашенного круга
- **square** – маркер в виде закрашенного квадрата

Пример 2

```
<UL>
<LI> <font color = red>красный;</font>
<LI> <font color = 900090000000>желтый;</font>
<LI> <font color = green>зеленый;</font>
</ul>
<UL type=disc>
<LI> <font color = red>красный;</font>
<LI> <font color = 900090000000>желтый;</font>
<LI> <font color = green>зеленый;</font>
</ul>
<UL type=square>
<LI> <font color = red>красный;</font>
<LI> <font color = 900090000000>желтый;</font>
<LI> <font color = green>зеленый;</font>
</ul>
```

После каждого пункта списка осуществляется перенос строки, причем без указания каких-либо структурных тегов. Если заголовок списка размещается между тегом *UL* и *LI*, то он будет отделен от начала самого списка одним переносом строки и отступом от левого края документа (как и все пункты в списках). При указании заголовка перед тегом *UL* браузер создает двойной перенос и размещает текст заголовка левее пунктов списка. Отличительной особенностью маркированных списков является возможность использования вместо стандартных HTML-маркеров свои собственные графические изображения. Для указания браузеру того, что в качестве маркера будет использовано графическое изображение, внутри тега *UL* вместо указателя элемента *LI* размещается обыкновенная HTML-конструкция описания графических объектов.

Пример 3

```
<UL>  
<IMG src = marker.gif> <font color = red>красный</font><br><br>  
<IMG src = marker.gif> <font color = 900090000000>желтый</font><br><br>  
<IMG src = marker.gif> <font color = green>зеленый</font><br><br>  
</UL>
```

Нумерованные списки

Нумерованный список часто называется упорядоченным и представляет собой определенную последовательность элементов. Нумерованные списки (Ordered List) отмечают тегами **... ** внутри которых располагаются пункты перечня информационных данных, за указание которых отвечает специальный тег *LI*. Чтобы программа правильно расставила номера вместо символов, выделяющих новый элемент, используются арабские или римские числа или буквы латинского алфавита.

Синтаксис:

```
<OL type=1 start=1 >  
<LI> элемент списка  
<LI> элемент списка </OL>
```

Тег составления нумерованных списков *OL* может включать атрибуты *type*, *start* и *compact*. Способ нумерации задается при помощи атрибута **type**.

Тип	Пояснение	Вид
type="1"	арабские цифры	1, 2, 3, ...
type="i"	римские цифры в нижнем регистре	i, ii, iii, ...
type="I"	римские цифры в верхнем регистре	I, II, III, ...
type="a"	латинские буквы нижнего регистра	a, b, c, ...
type="A"	латинские буквы верхнего регистра	A, B, C, ...

Типом нумерации, используемым по умолчанию, является *type="1"*, поэтому при создании нумерованных списков такого типа необязательно создавать значения атрибута *type* (сам перпметр также можно опустить).

Пример 4

```
<OL>
<li> красный;</li>
<li> желтый;</li>
<li> зеленый.</li>
</ol>
<OL type="I">
<li> красный;</li>
<li> желтый;</li>
<li> зеленый.</li>
</ol>
<OL type="A">
<li> красный;</li>
<li> желтый;</li>
<li> зеленый.</li>
</ol>
```

Возможно смешанное использование различных типов нумерации в пределах одного списка. Однако браузеры по-разному реагируют на такое смешение стилей.

Оборудование и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 7.

Задания

Задание 1. Создание нумерованных и нenumерованных списков.

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.

2. В элементе title укажите название странички "Задание 3. Создание нумерованных и нenumерованных списков на Web-странице по образцу".

3. Отформатируйте следующий текст согласно указаниям (даны в скобках курсивом).

Солнце должно быть: (Шрифт размером 4, красного цвета, выравнивание по центру)

1. *Теплым.*

2. *Круглым.*

3. *Желтым.*

Снег должен быть: (Шрифт размером 10, синего цвета, выравнивание по левому краю)

А. Белым.

В. Холодным.

С. Пушистым.

(Шрифт обоих, списков черный, размер 3).

Задание 2. Создайте маркированный список, содержащий фамилии ваших любимых актеров, создайте нумерованный список, перечисляющий цвета радуги, создайте список определений, содержащий название и описание породы собак.

Задание 3. Рецепт приготовления эликсира создайте в виде вложенного списка, применяя вложение маркированного списка в нумерованный.

1. Создать 3-уровневый список.

- уровень (верхний) - маркированный список стран;
- уровень (средний) - нумерованный список автомобильных фирм;
- уровень (нижний) - маркированный список автомобилей (количество элементов на каждом из уровней - не меньше 2).

Создать список определений (любой, количество элементов не меньше 5).

Создать нумеруемый список в виде меню (MENU).

2. Создать 3-уровневый список.

- уровень (верхний) - нумерованный список стран;
- уровень - маркированный список фирм, производящих принтеры;
- уровень - нумерованный список принтеров (количество элементов на каждом из уровней - не меньше 2).

Создать список определений (любой, количество элементов не меньше 5).

Создать нумерованный список в директории (DIR).

3. Создать 3-уровневый список.

- уровень (верхний) - маркированный список стран;
- уровень - нумерованный список фирм, производящих мониторы;
- уровень - нумерованный список мониторов (количество элементов на каждом из уровней - не меньше 2).

Создать список определений (любой, количество элементов не меньше 5).

Создать нумеруемый список в виде меню (MENU).

Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать: название дисциплины, название лабораторной работы, фамилию и инициалы студента, номер группы,

фамилию и инициалы преподавателя. Основная часть лабораторной работы должна содержать:

- ☐ вариант вашего задания;
- ☐ распечатку страниц выполненного задания;
- ☐ выводы по проделанной работе.

Защита работы

Защита работы заключается:

- ☐ в выполнении варианта своего задания и оформления отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- ☐ демонстрации преподавателю выполненного задания;
- ☐ в ответах на контрольные вопросы по лабораторной работе;
- ☐ в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Основы работы в Web-среде».

Контрольные вопросы

1. Перечислите основные типы списков в HTML-документах.
2. Охарактеризуйте маркированный (нумерованный) список.
3. Дайте характеристику нумерованных списков.
4. Дайте характеристику вложенным спискам.
5. Списки определений в HTML-документах.

Лабораторная работа 5

Тема: Вставка графических изображений

Цель работы: научиться размещать графические изображения на Web-странице.

Вставка графических изображений

Чтобы вставить изображение, нужно только иметь это самое изображение в формате *GIF* (файл с расширением *.gif) или *JPEG* (файл с расширением *.jpg или *.jpeg) и одну строчку в HTML-тексте.

**** - элемент для создания ссылки на графический файл (image). Он не содержит конечного тэга - вся необходимая информация задается при помощи атрибутов. Этот элемент является универсальным: с его помощью можно использовать изображения в гиперссылках, вставлять картинки в таблицы, просто размещать рисунки на Web-странице, решать задачи дизайна и т. д.

Необходимым атрибутом является **SRC** - указатель на графический файл:

SRC = "Ссылка на файл".

Допустим, нам нужно включить в документ изображение, записанное в файл picture.gif, находящееся в одном каталоге с HTML-документом. Тогда строчка будет вот такая:

Очень полезным атрибутом является **ALT**. Он позволяет выводить текст в тех местах, где должны располагаться рисунки. Страница может загружаться достаточно долго, и пока графические файлы на "подходе", пользователь должен видеть, какие изображения он сможет получить.

Пример 5

```
<HTML>
<HEAD>
<TITLE>Вставка изображения</title>
</head>
<BODY>
<IMG SRC="picture.gif" ALT="Картинка">
</body>
</html>
```

Встретив такую метку, браузер покажет на экране текст "Картинка" и начнет загружать на его место картинку из файла *picture.gif*. Атрибут **ALT**

может оказаться необходимым для старых браузеров, которые не поддерживают изображений, а также на случай, если у браузера отключена автоматическая загрузка изображений (при медленном подключении к Интернет это делается для экономии времени).

Файл, содержащий изображение, может находиться в другом каталоге или даже на другом сервере (обратите внимание на рисунок в строку адреса). В этом случае необходимо указать его полное имя (путь в файлу):

Оборудование и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 7.

Задания

Задание 1. Размещение графики на Web-странице

1. Создайте новую Web-страницу в редакторе **Блокнот**.
2. В элементе **<TITLE>** укажите название странички «*Задание 6. Размещение графики на Web-странице*».
3. Если в тэге **BODY** не указывать цвет странички, то по умолчанию фон будет белым. Сделайте цвет фона странички оранжевым.
4. Разместите на страничке любой рисунок. Для размещения изображения вам требуется указать путь к файлу в элементе **IMG**. Чтобы упростить описание пути к этому графическому файлу, скопируйте этот файл в ту же папку, в которой будет сохранена Web-страничка. Если файл находится не в одной папке с Web-страничкой, то необходимо указать относительный путь к этому файлу.
5. Сделайте подпись к рисунку с помощью атрибута **alt**.
6. Создайте рамку вокруг рисунка шириной 2 пиксела.

7. Над рисунком поместите заголовок самого большого размера и выровняйте его по центру страницы.

Задание 2. Изменение размеров изображения на Web-странице.

1. Создайте новую Web-страницу в редакторе **Блокнот**.
2. В элементе *TITLE* укажите название странички «Задание 7. Изменение размеров изображения на Web-странице».
3. Пусть фон странички будет зеленого цвета.
4. Определите размер выбранного вами изображения.
5. Вставьте это изображение на страничку, задав его размеры: ширина 100, высота 100 пикселей.
6. Просмотрите в браузере полученную страничку. Обратите внимание на следующие вещи:
 - В связи с тем, что вы непропорционально изменили размеры изображения, произошло искажение рисунка;
 - Так как вы увеличили размер изображения, произошла потеря качества рисунка (обратите внимание на края изображения);
 - Для размещения изображений на Web-страничке, лучше использовать прозрачный фон рисунка.

Задание 3. Использование рисунка в качестве гиперссылки

На страничках *page1.htm*, *page2.htm* и *page3.htm*, вместо текстовых гиперссылок используйте рисунки.

Для этого:

- Вставьте на первую страничку одно изображение и создайте гиперссылку для перехода на вторую страницу:

```
< A href = "page2.htm" IMG src = "*.gif" >... </a>
```

самостоятельно вставьте другое изображение для перехода с первой странички на третью;

- задайте бордюр обоим изображениям, равным 2;

- откройте в браузере первую страничку и, переходя по ссылкам со странички на страничку, убедитесь, что переходы работают верно.

- Самостоятельно вставьте изображения на страницы *page2.htm* и *page3.htm* и задайте соответствующие гиперссылки.

- Откройте в браузере первую страничку и, переходя по ссылкам со странички на страничку, убедитесь, что кнопки на всех страницах работают верно.

Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать: название дисциплины, название лабораторной работы, фамилию и инициалы студента, номер группы, фамилию и инициалы преподавателя. Основная часть лабораторной работы должна содержать:

- ☐ вариант вашего задания;
- ☐ распечатку страниц выполненного задания;
- ☐ выводы по проделанной работе.

Защита работы

Защита работы заключается:

- ☐ в выполнении варианта своего задания и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- ☐ демонстрации преподавателю выполненного задания;
- ☐ в ответах на контрольные вопросы по лабораторной работе;
- ☐ в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Основы работы в Web-среде».

Контрольные вопросы

1. Роль графики. Характеристика графических стандартов
2. Вставка графических изображений в HTML-документ.
3. Использование карт изображений

Лабораторная работа 6

Тема: Таблицы HTML-документов

Цель работы: познакомиться с основными приемами создания таблиц на языке гипертекстов, рассмотреть основные теги и атрибуты, научиться создавать, форматировать, заполнять таблицы, проектировать структуру таблицы.

Теоретическая часть

Одним из наиболее мощных и гибких средств представления информационных данных в HTML по праву являются таблицы.

Таблицы, изначально избранные в качестве визуального способа представления данных, сейчас имеют гораздо более важную функцию - управление структурой HTML-документа в целом. Удобство размещения данных в таблице неоспоримо, а преимущества перед другими средствами представления информации (например, списки) позволяют считать таблицы основополагающим структурным элементом любого HTML-документа.

Таблицы являются очень удобным средством форматирования данных на Web-странице. Основное удобство заключается в том, что браузер берет на себя заботу о прорисовке рамки таблицы. Размер рамки может быть автоматически согласован с размером окна просмотра в браузере и, разумеется, с размером находящихся в ячейках таблицы строк текста и рисунков. Кроме этого, таблицы позволяют решать чисто дизайнерские задачи: выравнивать части страницы друг относительно друга, размещать рядом рисунки и текст, управлять цветовым оформлением и т. д. Контурные таблицы показывать не обязательно, поэтому тег `<TABLE>` применим для размещения текстовых и графических компонент в разных местах страницы. При создании таблиц используется принцип вложения: внутри основного элемента таблицы (`TABLE`) создается ряд элементов, определяющих строки

(*TR*), а внутри этих элементов размещаются элементы для описания каждой ячейки в строке (*TD*, *TH*).

Основные методы создания таблиц

Для описания таблиц используется тег `<TABLE>`.

Тег `<TABLE>`, как и многие другие, автоматически переводит строку до и после таблицы.

Он позволяет задавать общие свойства таблицы и отделяет ее структуру от остальной части Web-страницы. Если начало и конец таблицы не определены, то никакие табличные команды не будут работать.

Рассмотрим параметры этого элемента.

Параметр align

Таблицу можно выравнивать по горизонтали при помощи атрибута *align*. Данный параметр устанавливает тип выравнивания всей таблицы относительно ширины HTML-документа.

- *align="left"* - влево;
- *align="center"* - по центру;
- *align="right"* - вправо.

Параметры width и height

Параметры *width* и *height* отвечают соответственно за ширину и высоту таблицы. Значение этих параметров может быть указано в процентах или пикселах:

Пример 1

```
<TABLE WIDTH="50%" HEIGHT="250">
```

Где *width = "50%"* задает таблицу шириной 50% страницы; *height = "250"* задает ширину таблицы, равной 250 пикселей.

В случае указания ширины или высоты в пикселах таблица будет иметь фиксированные размеры. Такая таблица при изменении размеров окна браузера останется неизменной. При использовании в качестве единицы измерения процентов ширина/высота таблицы будет варьироваться в зависимости от размеров окна браузера. В этом случае размер ячеек таблицы

будет уменьшаться/увеличиваться пропорционально заявленному значению в процентах. Содержимое таблицы будет перемещаться в окне браузера соответственно изменению его размеров.

Для управления видом рамки используются два атрибута. Дело в том, что браузер создает изображение рамки, имитируя ее трехмерность (выпуклость). На рамке можно различить фронтальную и боковую наклонную грани. Кроме этого, имитируется различие в их освещенности.

Параметр border

Параметр *border* задает ширину рамки. Если рамка не задана, то получим таблицу без рамок, тот же результат дает `<TABLE border="0">`. Также как и ширину можно задать в процентах и пикселях.

border = "Ширина в пикселях"

Параметр bordercolor

Параметр *bordercolor* предназначен для определения цвета рамки вокруг таблицы (поддерживается только Internet Explorer). Значение указывается в шестнадцатеричном формате или в виде наименования (например, *green*).

Параметр cellpadding

Параметр *cellpadding* задает расстояние между соседними ячейками таблицы по вертикали и горизонтали. Расстояние между соседними ячейками (как по вертикали, так и по горизонтали) всегда будет одинаково, т. к. HTML не позволяет назначать разные значения в пределах данного параметра.

cellpadding = Ширина в пикселях

Если значение этого атрибута равно нулю, рамка получается тонкой, заостренной. Если данный параметр не задан, по умолчанию задается величина, равная 2 пикселям. Атрибут полезен для размещения текста и графики именно там, где нужно.

Параметр cellspacing

Для всех ячеек таблицы можно задать размер пустого пространства между содержимым таблицы и ее границами. Для создания и регулирования

отступа между рамкой ячейки и ее содержимым (по вертикали и горизонтали) используется параметр *cellpadding*.

cellpadding = Число пикселей

Задание этого атрибута делает ячейки больше. Между рамкой таблиц и данными всегда сохраняется определенное расстояние. В некоторых случаях это позволяет улучшить восприятие таблицы, сделать текст в ячейках легко читаемым.

При значении параметра *cellpadding*, равном нулю, содержимое ячейки будет вплотную прижато к рамке таблицы, что ухудшит восприятие информации (это утверждение еще более актуально для таблиц с прозрачной рамкой, т. е. значением *border="0"*, - в этом случае текст соседних ячеек сольется друг с другом).

Группировка данных. Колонтитулы таблицы. Вложенные таблицы

Группировка данных

При построении таблиц мы можем легко задать единый тип выравнивания для отдельной ячейки и даже целого табличного ряда. Однако гораздо, чаще может возникнуть необходимость в едином выравнивании информационных данных конкретного столбца. *Столбец в таблице* - это последовательность ячеек, располагающихся в разных рядах.

Стандартными средствами HTML нам пришлось бы задавать один и тот же тип выравнивания для отдельно взятой ячейки, формирующей нужный столбец:

```
<TABLE>
<TR>
<TD align="right">Ячейка 1 с выравниванием вправо</td>
<TD align="center">Ячейка 1 с выравниванием по центру</td>
</tr>
<TR>
<TD align="right">Ячейка 2 с выравниванием вправо</td>
<TD align="center">Ячейка 2 с выравниванием по центру</td>
</tr>
</table>
```

При работе с браузером Internet Explorer, труд разработчика электронного документа может быть значительно облегчен за счет таких тегов, как `<col>` и `<colgroup>`.

Теги `<col>` и `<colgroup>` предназначены для определения свойств отображения табличных данных, сгруппированных по конкретному признаку. Они оба могут иметь параметр `span`, задающий количество соседних столбцов, и параметр `align`, становливающий единый тип выравнивания для выбранного столбца (возможные значения: по левому краю, по правому краю, по центру; формат записи аналогичен тегам `<TD>` и `<TH>`).

Помимо этого тег `<colgroup>` может содержать дополнительный параметр вертикального выравнивания данных - `valign` (возможные значения: по верхнему краю, по нижнему краю, по середине; формат записи аналогичен тегам `<TD>` и `<TH>`).

Разница между тегами `<col>` и `<colgroup>` заключается в условном объединении (группировке) взаимосвязанных данных отдельно взятого столбца таблицы.

Пример 12

```
<HTML>
<HEAD>
<TITLE>Группировка данных таблицы</title>
</head>
<BODY BGCOLOR="#FFFFFF" TEXT="black" LEFTMARGIN="40" RIGHTMARGIN="40"
MARGINWIDTH="40">
<TABLE align="center" border="2" cellspacing="0"
cellpadding="5" width="100%" height="200">
<COLGROUP align="center" span="2">
<COLGROUP align="right" span="2">
<TR>
<TD>Ячейка 1</td>
<TD>Ячейка 2</td>
<TD>Ячейка 3</td>
<TD>Ячейка 4</td>
</tr>
<TR>
<TD>Ячейка 5</td>
<TD>Ячейка 6</td>
<TD>Ячейка 7</td>
<TD>Ячейка 8</td>
</tr>
```

```

</table>
</body>
</html>

```

Ячейка 1	Ячейка 2	Ячейка 3	Ячейка 4
Ячейка 5	Ячейка 6	Ячейка 7	Ячейка 8

Рисунок 6. Результат работы пример 12

Оборудование и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 7.

Задания

Задание 1. Создание таблицы по образцу

- В элементе `<TITLE>` укажите название странички "Задание 1. Создание таблицы по образцу"
- Создайте таблицу, как показано на рисунок 8.

Один	Два	Три
Четыре	Пять	Шесть

Рисунок 8. Образец таблицы

- Текст в первой строке выровняйте по центру по горизонтали и по вертикали.
- Текст во второй строке выделите жирным шрифтом.

Задание 2. Создание таблицы по образцу.

- В элементе `<TITLE>` укажите название странички "Задание 2. Создание таблицы по образцу"
- Создайте HTML-документ, соответствующий следующему образцу, рисунок 9.

Примеры таблиц:

Стандартная таблица

Заголовок таблицы

Заголовок 1	Заголовок 2
Ячейка 1	Ячейка 2
Ячейка 3	Ячейка 4

Объединение ячеек

Заголовок 1	Заголовок 2	Заголовок 3
Ячейка 1		Ячейка 2
		Ячейка 3
		Ячейка 4
Ячейка 5		

Вложенная таблица

Ячейка 1.1.1	Ячейка 1.1.2	Ячейка 1.1.3	Ячейка 1.1.4	Ячейка 3.1.1	Ячейка 3.1.2	Ячейка 3.3.1
Ячейка 2.1.1 Ячейка 2.2.1				Ячейка 3.2.1	Ячейка 3.2.2	Ячейка 3.2.3
				Ячейка 4.1.1	Ячейка 4.1.2	
				Ячейка 4.2.1	Ячейка 4.2.2	
				Ячейка 4.3.1	Ячейка 4.3.2	

Табличка

Петровым 2 звонка

Рисунок 9. Образец таблицы

Задание 3. Форматирование таблицы

- Для выполнения этого задания воспользуйтесь результатами выполнения Задания 1.
- В элементе <TITLE> укажите название странички "Задание 3. Форматирование таблицы".
- Сделайте фон первого столбца зеленого цвета, второго - красного, третьего - синего.
- Задайте ширину бордюра таблицы, равным 3 пикселям.

- Ширину таблицы сделайте равной ширине экрана.
- Шрифт, которым написан текст внутри таблицы, сделайте белым.

Задание 4. Заполнение таблицы

- Для выполнения этого задания воспользуйтесь результатами выполнения Задания 1.
- В элементе <TITLE> укажите название странички "Задание 4. Заполнение таблицы".
- В первую ячейку первой строки вставьте рисунок из файла *.jpg.
- Во второй ячейке второй строки напишите ваше любимое четверостишие.
- Сделайте фон третьей ячейки второй строки красного цвета.

Задание 5.

Составьте выписку из своей зачетной книжки, оформив ее в виде таблицы.

Задание 6.

Изучите примеры и создайте WEB-страницу, на которой посередине разместите таблицу следующего вида (ширина таблицы=90%), рисунок 12.

Наиболее популярные модели струйных принтеров				
Изготовитель, название модели	Формат печати	Количество цветов	Макс. разрешение (dpi)	Макс. скорость печати (стр./мин.)
Hewlett Packard				
HP DeskJet 690C+	A4	6	600x600	5
HP DeskJet 970Cxi		4	2400x1200	12
Epson				
Epson Stylus Color 440	A4	4	720x720	4
Epson Stylus Color 900			1440x720	12

Рисунок 12. Образец таблицы

Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать: название дисциплины, название лабораторной работы, фамилию и инициалы студента, номер группы, фамилию и инициалы преподавателя. Основная часть лабораторной работы должна содержать:

- ☐ вариант вашего задания;
- ☐ распечатку страниц выполненного задания;
- ☐ выводы по проделанной работе.

Защита работы

Защита работы заключается:

- ☐ в выполнении варианта своего задания и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- ☐ демонстрации преподавателю выполненного задания;
- ☐ в ответах на контрольные вопросы по лабораторной работе;
- ☐ в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Основы работы в Web-среде».

Контрольные вопросы

1. Основные методы создания таблиц.
2. Параметры, которые используются при создании таблиц в HTML-документах.
3. Группировка данных
4. Колонтитулы таблицы
5. Прорисовка структуры таблицы
6. Вложенные таблицы
7. Создание цветных таблиц.

Лабораторная работа 7

Тема: Формы в HTML.

Цель работы: познакомиться с разными видами пользовательских форм

Теоретическая часть

Пользовательская форма – это совокупность стандартных HTML-конструкций ввода текстовой и прочей информации и программы-обработчика этой информации, работающей на Web-сервере. Иными словами, пользовательская форма (или HTML-форма) служит для передачи информационных данных серверу.

Результат конструкций языка разметки HTML интерпретируется браузером, с помощью которого пользователь электронного документа получает информацию. Таким образом, объединив все эти формулировки, можно сказать, что HTML-форма выступает в роли посредника между пользователем и сервером.

Посетитель Web-страницы вводит в HTML-форму определенные данные, которые обрабатываются программой и отсылаются на Web-сервер. Все эти действия укладываются в три стадии:

1. Ввод пользователем информации.
2. Обработка введенной информации программой, установленной на сервере.
3. Получение результата отправления введенной информации на Web-сервер (открытие нового HTML-документа, переадресация на предыдущую страницу и пр.).

В качестве программы-обработчика чаще всего выступает CGI-сценарий (скрипт, который обычно разрабатывается на языке Perl или C/C++ и который взаимодействует со специальным компонентом Web-сервера - Common Gateway Interface) или программы, написанные на основе таких серверных языков программирования, как *PHP*, *ASP*, *JSP* и др.

Значение пользовательских форм трудно переоценить – они являются особым средством HTML, дающим посетителю возможность не только пассивно просматривать информацию, но и быть задействованным в содержании Web-сайта. Такое свойство принято называть интерактивностью, которая на сегодняшний день встречается практически во всех электронных документах.

Диапазон функций, выполняемых HTML-формами, чрезвычайно широк, поэтому остановимся лишь на некоторых, наиболее распространенных направлениях.

Организация обратной связи

Форма обратной связи – это неотъемлемый атрибут Web-сайта любой тематики и функциональной направленности. Размещая электронный документ во Всемирной сети, мы уже подразумеваем реакцию пользовательской аудитории – похвалу и одобрение или же несогласие и критику.

В качестве средства обратной связи между пользователем и автором электронного документа могут выступать следующие.

1. **Почтовая форма.** Посетителя просят указать свои персональные данные и изложить суть обращения. Текст обращения обрабатывается программой и высылается автору сайта по электронной почте. При этом другие посетители сайта не могут получить доступ к содержанию отправленного сообщения;

2. **Гостевая книга.** Электронная книга жалоб и предложений, в которой любой желающий может оставить свой отзыв, поделиться впечатлениями о сайте, изложить критические замечания и указать на ошибки автора. В этом случае сообщения одного посетителя видимы для остальных пользователей гостевой книги;

3. **Интернет-форум.** Более мощный инструмент стимулирования оценки пользовательской аудитории, нежели почтовая форма или гостевая

книга. В интернет-форумах происходит многосторонний диалог, причем не только между человеком, задающим вопрос, и автором Web-сайта, но и между всеми пользователями данного интернет-ресурса. На вопрос одного пользователя может ответить другой, третий подскажет то, чего не упомянул второй, наконец, автор сайта подытожит высказывания всех пользователей и даст свой развернутый ответ. Часто в рамках интернет-форума работают дополнительные функциональные средства общения: личные сообщения, скрытые разделы форума для привилегированных пользователей, опросы общественного мнения и пр.;

4. **Чат.** Средство чата (от англ. "to chat" — разговаривать, болтать) подразумевает одновременное общение многих пользователей в режиме реального времени. В отличие от гостевой книги или форума, где ответ на поставленный вопрос может быть выслан и через час, и через день, общение в чате происходит между двумя (или более) собеседниками так, будто они сидят в одной комнате. Собеседники могут находиться в разных городах, государствах и временных зонах, однако, присутствуя в чате, все пользователи способны общаться одновременно. В чатах также могут присутствовать некоторые дополнительные функции: приваты (отдельный разговор между двумя пользователями), система почтовых сообщений, голосования и т. д.

Авторизация

Для получения доступа к информации на многих интернет-ресурсах существует необходимость регистрации в сервисе. После регистрации, которая подразумевает указание уникального пользовательского имени и пароля, пользователь должен пройти процедуру авторизации, т. е. проверки на наличие в базе данных сайта заявленного имени и пароля.

Система авторизации распространена на информационных ресурсах (новостные ленты, афиши событий), корпоративных серверах (закрытые

разделы для клиентов, партнеров, дистрибьюторов), сайтах электронной почты и т. д.

Проведение исследований

Для изучения пользовательского спроса на тот или иной вид услуг, выявления и последующего анализа реакции посетителей, на многих сайтах устраиваются всевозможные опросы, анкетирования, голосования, тесты и пр.

Набор стандартных HTML-конструкций облегчает участие пользователя в подобных исследованиях. Все, что ему потребуется, это отметить подходящие варианты ответа и подтвердить участие в опросе.

Пользовательский профиль

Регистрируясь на каком-либо интернет-ресурсе, человек указывает всякого рода информацию: имя, город и страну проживания, увлечения, место работы, контактные данные и пр. Совокупность таких данных составляет пользовательский профиль, который впоследствии возможно будет изменить путем редактирования соответствующих полей HTML-формы.

Структура пользовательских форм

Структура любой пользовательской формы описывается тегом-контейнером `<FORM>`, внутри которого могут располагаться теги и параметры составляющих элементов HTML-формы. В пределах тега `<FORM>` не может содержаться другая пользовательская форма. Закрывающий тег обязателен для указания. Тег `<FORM>` может содержать четыре основных параметра: *ACTION*, *METHOD*, *NAME* и *ENCTYPE*. Рассмотрим каждый из них.

`<FORM>...</form>`

Этот элемент необходим для построения сложных форм. Все формы начинаются тегом `<FORM>` и завершаются тегом `</form>`. После заполнения формы и подтверждения ввода со стороны пользователя введенная информа-

ция пересылается на сервер и обрабатывается при помощи CGI-программы, связанной с формой.

Параметр **action**

Параметр **action** является единственным обязательным параметром тега *<FORM>* и предназначен для указания пути на Web-сервере к программе обработчику данных пользовательской формы, например:

```
<form action = "http://www. название. домен/имя программы">
```

Путь к программе-обработчику может быть как относительный, так и абсолютный.

Значение этого параметра играет важную роль с точки зрения работоспособности HTML-формы. Если путь или название программы указаны неверно или же указанный файл не является исполняемым на стороне сервера, то обработка введенных пользователем данных может быть нарушена, произведена некорректно или вовсе не быть осуществленной.

Наиболее распространенными типами исполняемых на стороне сервера файлами являются CGI-программы (с расширениями *pi*, *cgi*, *fcgi*), PHP-скрипты (*php*, *php3*, *phtml*, *phtml*), ASP-приложения (*asp*), JSP-обработчики (*jsp*) и др.

Одним из вариантов обработки формы может быть пересылка данных по электронной почте:

```
<FORM ACTION ="mailto:Адрес@название.домен">
```

Оборудование и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 7.

Задание 1

№	Общие задания
1.	Создайте HTML форму "Личная", в которой пользователь будет задавать свое имя, фамилию, адрес, интересы (может быть несколько), username, password. Предусмотреть текстовое поле для комментария из нескольких строк. Следует учитывать следующие моменты компоновки и дизайна

	HTML форм выбор даты, месяца и года рождения должен производиться различными способами. Предусмотрите в данной форме гиперссылку для перехода на другой Web документ.
2.	В новом HTML документе разработайте HTML-форму, в которой на примере своей зачетной книжке (дисциплина и оценка), покажите использование различных видов списков для компоновки форм, использование флажков и переключателей.

Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать: название дисциплины, название лабораторной работы, фамилию и инициалы студента, номер группы, фамилию и инициалы преподавателя. Основная часть лабораторной работы должна содержать:

- ☐ вариант вашего задания;
- ☐ распечатку страниц выполненного задания;
- ☐ выводы по проделанной работе.

Защита работы

Защита работы заключается:

- ☐ в выполнении варианта своего задания и оформления отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- ☐ демонстрации преподавателю выполненного задания;
- ☐ в ответах на контрольные вопросы по лабораторной работе;
- ☐ в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Основы работы в Web-среде».

Контрольные вопросы

1. Перечислите основные теги для создания HTML-форм.
2. Тег (дескриптор) FORM.
3. Ввод информации в HTML-форму. Тег INPUT.
4. Создание паролей в HTML-формах.
5. Осуществление выбора информации с помощью флажков и переключателей.

6. Назначение кнопки RESET.
7. Назначение кнопки SUBMIT.
8. Выбор информации в HTML-форме. Основные теги и их характеристика.
9. Характеристика дескриптора TEXTAREA.
10. Какой тег следует указывать при поиске из нескольких массивов информации?
11. Если форма, проектируемая пользователем достаточно велика, то какой способ используют для размещения информации?
12. Какой тег используется для формирования горизонтальной строки?
13. Какой тег используют для перевода строки?
14. Что улучшает совместное использование в Web документе таблиц и форм?
15. Какие элементы управления позволяют передать данные с компьютера клиента на сервер?
16. Напишите пример одновременного использования HTML-форм, таблиц и кнопки передачи запроса.

Лабораторная работа 8

Тема: Технология CSS

Цель работы: познакомиться с технологией CSS

Назначение CSS

Дизайн Web-узлов - это точное размещение компонентов HTML-страниц относительно друг друга в рабочей области окна браузера.

Неточность данного определения Web-дизайна очевидна. В нем не учтены ни цвет, ни форма, ни другие свойства компонентов HTML-страниц. Главное в этом определении - показать ограниченность возможностей

HTML-разметки. Позиционирование компонентов на странице является одним из самых слабых мест в HTML.

Компонентами страницы являются: блоки текста, графика и встроенные в страницу приложения. Размер и границы каждого из этих компонентов в рамках HTML-разметки задаются с разной степенью точности. Размер графики и приложений можно задать с точностью до пикселя. Размеры текстовых блоков в HTML задать нельзя. Они вычисляются браузером на основе относительного размера шрифта умолчания.

Автор страницы не может заранее определить настроек браузера пользователя, что существенно ограничивает число вариантов представления информации на странице.

Нельзя сказать, что разработчики браузеров не предпринимали попыток изменить данную ситуацию. В ранних версиях браузеров CERN для платформы NEXT и браузерах WWWC автору страницы давалась возможность переопределять настройки умолчания браузера через HTML-разметку. Но этот подход не получил продолжения в коммерческих продуктах и постепенно завял.

Другой способ управления настройками браузера - программирование на JavaScript. Бурное развитие этого языка заставляет говорить о возможности полного контроля над процессом отображения HTML-страниц. Недостаток JavaScript - отказ от декларативного характера разметки и относительно большой объем кода для переопределения свойств элементов разметки.

Спецификация **CSS** (Cascading Style Sheets) позволяет остаться в рамках декларативного характера разметки страницы и дает полный контроль над формой представления элементов HTML-разметки.

Каскадные таблицы стилей призваны разрешить противоречие между точностью определения размеров картинок и приложений с одной стороны, и точностью определения размеров блоков текста и его начертания с другой.

Кроме размера компонентов, таблицы стилей позволяют определить цвет и начертание текстового фрагмента, изменять эти параметры внутри текстового блока, выполнять выравнивание текстового блока относительно других блоков и компонентов страницы.

Наличие всех этих возможностей позволяет говорить о CSS как о средстве разделения логической структуры документа и формы его представления. Логическая структура документа определяется элементами HTML-разметки, в то время как форма представления каждого из этих элементов задается CSS-описателем элемента.

CSS позволяет полностью переопределить форму представления элемента разметки по умолчанию. Например, `<i>...</i>` определяет отображение текста курсивом:

`<i> Отообразим текст курсивом</i>`

Отообразим текст курсивом

А теперь переопределим стиль отображения для элемента разметки `i`:

`<i style="text-decoration:underline;font-style:normal;">`

Отообразим текст курсивом

`</i>`

Этот пример показывает, что привычный стиль отображения элементов может быть полностью изменен при помощи CSS. В данной технологии HTML-разметка носит чисто декларативный характер.

Практическая значимость CSS для Web-инжиниринга (совокупности технологий разработки и сопровождения Web-узлов) заключена в том, что процесс создания узла можно формализовать и представить в виде следующей последовательности действий:

- Сначала нужно определиться с номенклатурой страниц, т.е. все страницы проектируемого Web-узла разбить на типы, например,,: домашняя страница, навигационные страницы, информационные страницы,

коммуникационные страницы и т.п.. У каждого узла этот перечень может быть своим.

- Для каждого из типов страниц разрабатывается определенная логическая структура (стандартный набор компонентов страницы).

- После этого разрабатывается навигационная карта узла и форма ее реализации на страницах.

- Для каждого стандартного компонента страницы разрабатывается стиль его отображения (CSS-описатель).

- Теперь остается только рисовать картинки, создавать анимацию, писать программы, вручную вводить текст и графику или генерировать содержание страниц автоматически во время обращения к ним.

Объяснив таким образом роль и назначение CSS среди многообразия Web-технологий, мы перейдем непосредственно к обсуждению практики применения каскадных таблиц стилей.

Оборудование и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 7.

Задание 1

Создайте стилевой файл содержащий оформление web-сайта. Обязательно определение оформления следующих элементов:

- Заголовки 1-4 уровней.
- Пункты меню 1-4 уровней.
- Гиперссылки.
- Таблицы и ячейки таблиц.
- Основной текст.

Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать: название дисциплины, название

лабораторной работы, фамилию и инициалы студента, номер группы, фамилию и инициалы преподавателя. Основная часть лабораторной работы должна содержать:

- ☐ вариант вашего задания;
- ☐ распечатку страниц выполненного задания;
- ☐ выводы по проделанной работе.

Защита работы

Защита работы заключается:

- ☐ в выполнении варианта своего задания и оформления отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- ☐ демонстрации преподавателю выполненного задания;
- ☐ в ответах на контрольные вопросы по лабораторной работе;
- ☐ в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Основы работы в Web-среде».

Контрольные вопросы

17. Перечислите основные теги для создания HTML-форм.
18. Тег (дескриптор) FORM.
19. Ввод информации в HTML-форму. Тег INPUT.
20. Создание паролей в HTML-формах.
21. Осуществление выбора информации с помощью флажков и переключателей.
22. Назначение кнопки RESET.
23. Назначение кнопки SUBMIT.
24. Выбор информации в HTML-форме. Основные теги и их характеристика.
25. Характеристика дескриптора TEXTAREA.
26. Какой тег следует указывать при поиске из нескольких массивов информации?

27. Если форма, проектируемая пользователем достаточно велика, то какой способ используют для размещения информации?

28. Какой тег используется для формирования горизонтальной строки?

29. Какой тег используют для перевода строки?

30. Что улучшает совместное использование в Web документе таблиц и форм?

31. Какие элементы управления позволяют передать данные с компьютера клиента на сервер?

32. Напишите пример одновременного использования HTML-форм, таблиц и кнопки передачи запроса.

Лабораторная работа 9

Тема: Внедрение объектов JavaScript в HTML-документ

Цель работы: изучить способы внедрения объектов JavaScript в HTML-документ

Теоретическая часть

JavaScript был разработан совместно компаниями Sun Microsystems и Netscape. За синтаксическую основу нового языка был взят язык Java, в свое время разработанный компанией Sun Microsystems. В последнее время популярность JS очень возросла в результате выхода в свет браузеров, поддерживающих данный язык. JS - интерпретатор с элементами объектно-ориентированной модели. Хотя он и лишен возможностей создания собственных классов, но он оперирует стандартными объектами. Так как обработчик находится на компьютере пользователя, JS, будучи интерпретатором, использует методы и свойства объектов обозревателя на пользовательском компьютере. JS имеет возможность написания пользовательских функций, имеет ряд операторов, но работает с объектами, их методами, свойствами и событиями. Также имеется иерархия наследования свойств объектов. Сложность составляет и то, что JS встраивается в html документ и взаимодействует с ним.

Microsoft выпустила похожие версии языка под названием JScript, поэтому под названием "JavaScript" часто понимается любая версия языка, в том числе и Microsoft JScript.

В большинстве случаев при упоминании JavaScript подразумевается так называемый клиентский JavaScript, интерпретатор которого встроен в Web-браузеры. Однако JavaScript изначально был разработан как универсальный язык программирования для встраивания в любое приложение и обеспечения возможности написания в нем сценариев. Например, ActionScript, язык сценариев, доступный в Macromedia Flash, также смоделирован в соответствии со стандартом ECMAScript.

Рассмотрим способы использования JavaScript внутри HTML-документа.

Способ первый

На первом этапе мы составим программу JavaScript, которая вставляет слова "Hello, world!" непосредственно в документ HTML. Программы или сценарии JavaScript встраиваются в документ HTML. Посмотрите на пример 1, в которой приведен исходный текст документа с программой, составленной на JavaScript.

Пример 1

```
<HTML>
<HEAD>
  <TITLE>Hello, world!</TITLE>
</HEAD>
<BODY>
  <H1>JavaScript Test</H1>
  <SCRIPT >
    <!--
      document.write("Hello, world!");
    // -->
  </SCRIPT>
</BODY>
</HTML>
```

Как и подобает любому документу HTML, он ограничен операторами `<HTML>`, `</HTML>` и состоит из двух разделов. Раздел заголовка выделяется операторами `<HEAD>` и `</HEAD>`, а раздел тела документа - операторами `<BODY>` и `</BODY>`. Программа JavaScript этом примере встроена в тело документа HTML при помощи операторов `<SCRIPT>` и `</SCRIPT>`.

С помощью оператора `<SCRIPT>` можно встроить в документ сценарий, составленный на языке JavaScript или VBScript. Язык указывается с помощью параметра `LANGUAGE`. Текст сценария оформлен как комментарий с применением операторов `<!--` и `-->`. Это сделано для того, чтобы сценарий не вызывал проблем у пользователей, браузеры которых не могут работать с JavaScript. Такие браузеры просто проигнорируют сценарий. Обратите внимание на строку, которой завершается комментарий: `// -->` Перед символами `-->` записаны два символа `/`. Это позволяет обеспечить

работоспособность сценария в различных браузерах. Некоторые из них (например, Netscape Navigator) в сценариях JavaScript рассматривают строку -> как ошибочную. Символы // используются в JavaScript для выделения комментариев и предписывают браузерам игнорировать символы, записанные после них (в том числе и -->). Для обозначения комментариев можно использовать также конструкцию /*...*/. Этот способ удобен, если комментарий содержит несколько строк.

Наша первая программа весьма проста и содержит только одну строку:

```
document.write("Hello, world!");
```

Здесь для объекта с именем document вызывается метод write. В качестве параметра ему передается текстовая строка "Hello, world!". Строка закрывается символом точка с запятой, хотя этот символ может и отсутствовать. Объект document - это документ HTML, загруженный в окно браузера. Он содержит в себе объекты, свойства и методы, предназначенные для работы с элементами этого документа HTML, а также для взаимодействия с другими объектами. Метод write записывает в тело документа HTML приветственную строку "Hello, world!". При этом документ будет выглядеть так, как будто эта строка находится в нем на месте сценария:

```
<HTML>
<HEAD>
<TITLE>Hello, world!</TITLE>
</HEAD>
<BODY>
<H1>JavaScript Test</H1>
Hello, world!
</BODY>
</HTML>
```

Способ второй

Исходный текст любого сценария должен включаться в документы HTML. Однако, есть техническая возможность оформлять программы на JavaScript в отдельных файлах, а в страницах HTML указывать на эти файлы ссылки. Браузер, загружая оформленные подобным образом HTML

документы, загружает оформленные в отдельных файлах сценарии и подставляет их вместо соответствующих ссылок. Такой способ включения JavaScript сценариев удобен, если один и тот же сценарий должен быть включен во множество документов HTML, или же если есть необходимость скрыть исходный код от просмотра пользователями (через просмотр источника).

Ссылки на файлы с подгружаемыми скриптами оформляются с помощью параметра SRC тега <SCRIPT>, допускающего указывать адрес URL файла сценария. Следующий пример демонстрирует использование параметра SRC. В примере 2 находится исходный текст документа HTML, содержащий ссылку на файл сценария hello.js.

Пример 2

```
<HTML>
<HEAD>
  <TITLE>Hello, world!</TITLE>
</HEAD>
<BODY>
  <H1>JavaScript Test</H1>
  <SCRIPT SRC="hello.js">
</SCRIPT>
</BODY>
</HTML>
```

Ссылка оформлена с применением операторов <SCRIPT> и </SCRIPT>, однако между этими операторами нет ни одной строчки исходного текста. Этот текст перенесен в файл hello.js (пример 3).

Пример 3 Файл hello.js

```
document.write("<HR>");
document.write("Hello, world!");
document.write("<HR>");
```

В параметре SRC вышеприведенного примера задано только имя файла, так как он находится в том же каталоге, что и ссылающийся на него файл документа HTML. Однако можно указать и относительный путь, и полный адрес URL, например: <SCRIPT LANGUAGE="JavaScript"

SRC="http://www.myserver.ru/scripts/hello.js"> Существенно, чтобы файл, в котором находится исходный текст сценария JavaScript, имел тип js. В противном случае сценарий работать не будет.

Оборудование и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 7.

Задания

1. Измените последний пример так, чтобы диалоговая панель возникала не при наведении курсора мыши, а при выборе ссылки (событие onClick).
2. Измените пример так, чтобы при наведении курсора мыши на ссылку, выполнялась бы процедура, выводящая в окно браузера фразу "Hello, world!".
3. Напишите скрипт, который сначала выводит на экран диалоговое окно, а затем, после нажатия кнопки "ОК", в окне браузера пишет фразу "Hello, world!".
4. Напишите скрипт, который запрашивает у пользователя информацию, а затем выводит ее в диалоговом окне.
5. Составьте документ так, чтобы диалоговое окно для ввода информации предлагалось только после наведения курсора мыши на ссылку, и введенная пользователем текстовая строка выводилась бы в виде диалогового окна, или в окно браузера.

Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать: название дисциплины, название лабораторной работы, фамилию и инициалы студента, номер группы, фамилию и инициалы преподавателя. Основная часть лабораторной работы должна содержать:

- ☐ вариант вашего задания;

- ☐ распечатку страниц выполненного задания;
- ☐ выводы по проделанной работе.

Защита работы

Защита работы заключается:

- ☐ в выполнении варианта своего задания и оформления отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- ☐ демонстрации преподавателю выполненного задания;
- ☐ в ответах на контрольные вопросы по лабораторной работе;
- ☐ в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Основы работы в Web-среде».

Контрольные вопросы

1. Расскажите способы использования JavaScript внутри HTML-документа.
2. Использование JavaScript внутри HTML-документа с помощью оператора <SCRIPT>
3. Поясните, как оформить сценарии в отдельном файле.

Лабораторная работа 10

Тема: Использование базовых конструкций языка JavaScript

Цель работы: изучить базовые конструкции языка JavaScript

Теоретическая часть

Условные операторы

В языке JavaScript предусмотрен условный оператор if-else, который позволяет выполнять разные программные строки в зависимости от условия.

Общий вид оператора if-else представлен ниже:

```
if(условие)
    строка 1
[else
    строка 2]
```

Часть оператора, выделенная квадратными скобками, является необязательной.

Существует также специальный тип условного оператора, который называется оператором `?:`. Этот оператор в общем виде записывается так:

выражение ? строка 1 : строка 2

При вычислении оператора `?:` вначале оценивается логическое выражение, расположенное в левой части. Если оно равно `true`, выполняется строка 1, а если `false` - строка 2.

Ниже приведен пример использования условного оператора `?:` для присвоения значения переменной `bAccessDenied` в зависимости от содержимого переменной `nYourAge`:

```
bAccessDenied = (nYourAge < 18 || nYourAge > 99) ? true : false;
```

Операторы цикла

В языке JavaScript есть несколько операторов, предназначенных для организации циклов.

Оператор `for`

Общий вид оператора `for` представлен ниже:

```
for([инициализация;] [условие;] [итерация])  
{  
  ...  
  строки тела цикла  
  ...  
}
```

В области инициализации обычно выполняется присваивание начальных значений переменным цикла. Здесь допустимо объявление новых переменных при помощи ключевого слова `var`. Вторая область задает условие выхода из цикла. Это условие оценивается каждый раз при прохождении цикла. Если в результате оценки получается логическое значение `true`, выполняются строки тела цикла. Область итерации применяется для изменения значений переменных цикла, например, для увеличения счетчика цикла.

Оператор `for-in`

Оператор for-in предназначен для просмотра всех свойств объекта и записывается в следующем виде:

```
for(переменная in объект)
{
    ...
    строки тела цикла
    ...
}
```

Оператор while

Для организации итерационных циклов с предусловием используется оператор while:

```
while(условие)
{
    ...
    строки тела цикла
    ...
}
```

Если в результате оценки условия получается значение true, тогда итерация выполняется, если false - цикл прерывается.

Оператор break

С помощью оператора break можно прервать выполнение цикла, созданного операторами for или while, в любом месте.

Пример 1

```
var i = 0;
while(true)
{
    ...
    i++;
    if(i > 10)
        break;
    ...
}
```

Оператор continue

Выполнение оператора continue внутри цикла for или while приводит к тому, что итерация прерывается, а затем возобновляется заново. Этот оператор не прерывает цикл. Ниже приведен пример использования оператора continue:

```
var i = 0;
while(i < 100)
```

```
{  
    i++;  
    if(i < 10)  
        continue;  
    ...  
}
```

Здесь фрагмент тела цикла, отмеченный многоточием, будет выполняться только после того, как значение переменной *i* станет равным 10. Когда же это значение достигнет 100, цикл будет завершен.

Оборудование и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 7.

Задания

1. Напишите скрипт, который запрашивает два числа и выводит в окне браузера наибольшее из них.

2. Напишите скрипт, который запрашивает номер месяца и выводит название времени года.

3. Напишите скрипт, который отображает диалоговое окно с кнопками "Ок" и "Отмена" (функция `confirm`) и выводит в окно браузера сообщение о том, какая из кнопок была нажата (используйте конструкцию `if(confirm("Сообщение")) ...`)

4. Напишите скрипт, запрашивающий количество учащихся студенческой группы и в соответствии с этим количеством - их фамилии и имена. Фамилии и имена необходимо отобразить в окне браузера.

5. Напишите скрипт, который запрашивает фамилии и имена до тех пор, пока пользователь в окне ввода не нажмет кнопку "Отмена". Фамилии и имена необходимо отобразить в окне браузера.

6. Напишите скрипт, выводящий в окно браузера таблицу умножения

7. Создайте скрипт, который выводил бы на странице горизонтальные линии разной длины;

8. Создайте скрипт, который выводил бы на странице таблицу умножения;

9. Создайте скрипт, который выводил бы на странице таблицу перевода температуры из градусов по шкале Цельсия (с) в градусы по шкале Фаренгейта (f) по формуле $f = 1.8c + 32$ для значений температуры от 0 до 30 °C с шагом 1 °C;

10. Создайте скрипт, который выводил бы на странице таблицу соответствия между весом в фунтах и весом в килограммах для значений от 1 до 30 фунтов с шагом 1, считая, что 1 фунт=0,4 кг.

Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать: название дисциплины, название лабораторной работы, фамилию и инициалы студента, номер группы, фамилию и инициалы преподавателя. Основная часть лабораторной работы должна содержать:

- ☐ вариант вашего задания;
- ☐ распечатку страниц выполненного задания;
- ☐ выводы по проделанной работе.

Защита работы

Защита работы заключается:

- ☐ в выполнении варианта своего задания и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- ☐ демонстрации преподавателю выполненного задания;
- ☐ в ответах на контрольные вопросы по лабораторной работе;
- ☐ в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Основы работы в Web-среде».

Контрольные вопросы

1. Условный оператор if-else языка JavaScript
2. Оператор for языка JavaScript.
3. Оператор for-in языка JavaScript.
4. Оператор while языка JavaScript.

5. Оператор break языка JavaScript.
6. Оператор continue языка JavaScript.

Лабораторная работа 11. Функции. Рекурсивные функции

Цель работы: научиться разрабатывать скрипты на языке JavaScript с использованием функций и рекурсивных функций.

Теоретический материал по теме

Функция - это программный код, который определяется один раз и затем может вы-зываться на выполнение любое количество раз. В JavaScript **функция** - специальный тип объектов, позволяющий формализовать средствами языка определённую логику поведе-ния и обработки данных.

Объявление функции осуществляется следующим образом:

```
function имя(параметры){  
  
    код функции  
  
}
```

Имя - обязательный идентификатор, определяющий **имя функции**. Имя функции мо-жет содержать буквы, цифры (0-9), знаки «\$» и «_»; в качестве букв рекомендуется ис-пользовать только буквы английского алфавита (a-z, A-Z); имя функции, также как и имя переменной, не может начинаться с цифры.

В качестве имени функции обычно выбирают глагол, т. к. функция выполняет дей-ствие. Оно должно быть простым, точным и описывать действие функции, чтобы про-граммист, который будет читать код, получил верное представление о том, что делает функция. Как правило, используются глагольные префиксы, обозначающие общий харак-тер действия, после

которых следует уточнение. Обычно в командах разработчиков действуют соглашения, касающиеся значений этих префиксов. Например, функции, начинающиеся с...

"show..." – что-то

показывают; "get..." –

возвращают значение;

"calc..." – что-то

вычисляют; "create..." –

что-то создают;

"check..." – что-то проверяют и возвращают логическое значение, и т.д.

Параметры функции - это локальные переменные функции, которые определяются на этапе объявления функции в круглых скобках. Обратиться к параметрам можно только внутри функции, снаружи они не доступны. Параметров может быть указано любое количество; если параметров несколько, то их между собой необходимо разделить посредством запятой. Параметры позволяют более удобно (по имени) получить переданные аргументы функции при её вызове.

При вызове функции, ей могут передаваться значения, которыми будут инициализированы параметры. Значения, которые передаются при вызове функции, называются **аргументами**. Аргументы, указываются через запятую. Когда при вызове функции ей передаётся список аргументов, эти аргументы присваиваются параметрам функции в том порядке, в каком они указаны: первый аргумент присваивается первому параметру, второй аргумент – второму параметру и т. д.

Код функции - это набор инструкций, заключенный в фигурные скобки, которые необходимо выполнить при её вызове. Код функции называют ещё

её **телом**. Тело функции может быть пустым, но фигурные скобки должны быть указаны всегда.

С помощью инструкции `return` функция может вернуть некоторое значение (результат работы функции) программе, которая её вызвала. Возвращаемое значение передаётся в точку вызова функции. Инструкция `return` имеет следующий синтаксис:

`return выражение;`

В программу возвращается не само выражение, а результат его вычисления. Инструкция `return` может быть расположена в любом месте функции. Как только будет достигнута инструкция `return`, функция возвращает значение и немедленно завершает своё выполнение. Код, расположенный после инструкции `return`, будет проигнорирован. Если инструкция `return` не указана или не указано возвращаемое значение, то функция вернёт значение `undefined`.

Примеры:

1)

```
function sum(a, b) { var s = a
    + b; alert(s);
}
sum(5, 7);           // 12
```

2)

```
function calc(a) {
    return a * a;
}
var x = calc(5);
alert(x);           // 25
```

3)

```
function check(a, b) {
    if(a > b) return a;
    else return b;
}
alert(check(3, 5)); // 5
```

Рекурсивно определенная функция - это функция, которая определяется в терминах более простой версии самого себя. **Рекурсия** – это

приём программирования, полезный в ситуациях, когда задача может быть естественно разделена на несколько аналогичных, но более простых задач. Или когда задача может быть упрощена до несложных действий плюс простой вариант той же задачи.

Рекурсивная функция обязательно должна иметь условие завершения, если его не указать, функция будет вызываться до тех пор, пока не будет достигнута максимальная глубина рекурсии, затем будет сгенерировано исключение. Общее количество вложенных вызовов называют **глубиной рекурсии**. Максимальная глубина рекурсии в браузерах ограничена 10 000 рекурсивных вызовов.

Пример рекурсивной функции, определяющей факториал числа n!:

```
function getFactorial(n){  
  if (n === 1) return 1;  
  else return n*getFactorial(n-1);  
}  
var result = getFactorial(4);  
alert(result); // 24
```

Функция getFactorial() возвращает значение 1, если параметр n равен 1, либо возвращает в качестве результата произведение параметра n на значение той же функции getFactorial, в которую передается значение n-1. Например, при передаче числа 4, у нас образуется следующая цепочка вызовов:

```
var result = 4 * getFactorial(3); var result = 4 *  
3 * getFactorial(2);  
var result = 4 * 3 * 2 * getFactorial(1); var result =  
4 * 3 * 2 * 1; // 24
```

Задания для выполнения

Задание 1. Разработайте скрипт, реализующий алгоритм решения задачи с использованием функции. Ввод исходных данных осуществлять с использованием функции `prompt`. Вывод полученных результатов осуществлять с использованием функции `alert`. Сохраните скрипт в файле **script_3_1.js**.

Задание 2. Разработайте скрипт, реализующий алгоритм решения задачи с использованием рекурсивной функции. Ввод исходных данных осуществлять с использованием функции `prompt`. Вывод полученных результатов осуществлять с использованием функции `alert`. Сохраните скрипт в файле **script_3_2.js**

Задание 3. Создайте по образцу html-документы **lab_3_1.html** и **lab_3_2.html** и подключите к ним внешние скрипты **script_3_1.js** и **script_3_2.js** соответственно. Для тестирования работоспособности каждого скрипта разработайте по 3 теста (включающих наборы входных и выходных данных). Выполните тестирование и отладку скриптов в браузере.

Варианты индивидуальных задач к Заданию 1:

1. Написать и протестировать функцию, которая вычисляет сумму квадратов n ее аргументов, где $1 \leq n \leq 3$.
2. Даны действительные числа x, y и z . Получить $u = \frac{\max(x, y) + \max(x + y, z)}{(\max(0.5, x + z))^2}$.
3. Даны действительные числа s, t . Получить $f(t, -2s, 1.17) + f(2.2, t, s - t)$, где $f(a, b, c) = \frac{2a - b - \sin c}{5 + |c|}$.
4. Даны действительные числа a, b, c . Получить $\frac{\max(a, a + b) + \max(a, b + c)}{1 + \max(a + bc, 1, 1.5)}$.
5. Написать и протестировать функцию, определяющую периметр плоского n -угольника, координаты вершин которого передаются в качестве параметра в виде $2n$ чисел: x_i и y_i .
6. Написать и протестировать функцию и именем COSIN, вычисляющую $\cos(x + iy)$ по формуле $\cos(x + iy) = \cos x \cdot \frac{e^y + e^{-y}}{2} \cdot \sin x \cdot \frac{e^y - e^{-y}}{2} i$, где x и y — два параметра вещественного типа.
7. Даны действительные числа a и b . Получить $u = f(0.5, a) + f(a, b) + f(a + b, a - b)$, где $f(x, y) = \frac{x^2 + xy - y^2}{1 + x^2 + y^2}$.

8. Написать и протестировать функцию, которая возвращает следующие значения:

$$y = \begin{cases} 1 + \sqrt{1+x^2}, & \text{если } x < 0, \\ 0, & \text{если } x = 0, \\ 1 - \sqrt{1+x^2}, & \text{если } x > 0. \end{cases}$$

9. Написать и протестировать функцию и именем *EXPONENT*, вычисляющую $e^{x+iy} = e^x(\cos y + i \sin y)$, где x и y — два параметра вещественного типа.

10. Дано действительное число y . Получить $\frac{1.7t(0.25) + 2t(1+y)}{6 - t(y^2 - 1)}$, где $t(x) = \frac{\sum_{k=0}^{10} \frac{e^{2k+1}}{(2k+1)!}}{\sum_{k=0}^{10} \frac{e^{2k}}{(2k)!}}$.

11. Написать и протестировать логическую функцию, которая определяет, принадлежит ли заданная точка к внутренней области круга $(x - x_0)^2 + (y - y_0)^2 \leq R^2$. Формальные параметры: X_0, Y_0 — координаты центра круга, R — радиус круга, x, y — координаты проверяемой точки.

12. Даны действительные числа m и n . Получить $l = \frac{m! + n!}{(m+n)!}$, введя в употребление функцию $fact(n) = 1 \cdot 2 \cdot \dots \cdot n$.

13. Даны натуральное число n , действительные числа $s, t, a_0, \dots, a_n$. Получить $p(1) - p(t) + p^2(s - t) - p^3(1)$, где $p(x) = a_n e^{nx} + a_{n-1} e^{(n-1)x} + \dots + a_0$.

14. Написать и протестировать функцию, вычисляющую расстояние между двумя точками в n -мерном евклидовом пространстве, где $1 \leq n \leq 3$. Координаты точек передаются в виде $2n$ отдельных параметров.

15. Даны натуральные числа k, l, m , действительные числа $x_1, \dots, x_k, y_1, \dots, y_l, z_1, \dots, z_m$. Получить:

$$t = \begin{cases} (\max(y_1, \dots, y_l) + \max(z_1, \dots, z_m)) & \text{при } \max(x_1, \dots, x_k) \geq 0, \\ 1 + (\max(x_1, \dots, x_k))^2 & \text{при } \max(x_1, \dots, x_k) < 0. \end{cases}$$

16. Написать и протестировать функцию для вычисления t с формальным параметром x . $f = -0.5 + 1.25x - 3.25x^2 - 1.87x^3 + 0.5x^4$. Формальный параметр и значение функции должны быть действительного типа.

17. Даны действительные числа a, b . Получить $u = \min(a, b)$, $v = \min(ab, a+b)$, $\min(u+v, 3.25)$.

18. Написать и протестировать функцию для вычисления H с формальными параметрами x, y и z : $H = (x+4)^3 - \frac{y}{2} + x \ln z$. Формальные параметры и значение функции должны быть действительного типа.

19. Даны действительные числа s, t . Получить $h(s, t) + \max(h^2(s - t, st), h^4(s - t, s + t)) + h(1, 1)$, где $h(a, b) = \frac{a}{1+b^2} + \frac{b}{1+a^2} - ab$.
20. Написать и протестировать функцию для вычисления g с формальными параметрами x, y $g = e^{-(x-1)^2} - y(x-1)^2$. Формальные параметры и значение функции должны быть действительного типа.
21. Даны действительные числа s, t . Получить $g(1, 2, s) + g(t, s) - g(2s - 1, st)$, где $g(a, b) = \frac{a^2 + b^2}{a^2 + 2ab + 3b^2 + 4}$.
22. Написать и протестировать функцию, позволяющую найти корень m -той степени из x , применив тождество: $\sqrt[m]{x} = e^{\frac{\ln x}{m}}$.
23. Написать и протестировать функцию для вычисления I с формальным параметром x : $I = e^M - e^{-M}$, где $M = x^3 + 0.5x^2 - x$.
24. Написать и протестировать функцию f , которая возвращала бы среднее геометрическое n ее параметров по формуле $f = \sqrt[n]{x_1 \cdot x_2 \cdot \dots \cdot x_n}$.
25. Написать и протестировать функцию, которая одной действительной переменной присваивает значение, равное сумме квадратов двух действительных значений, а другой переменной — значение, равное удвоенному произведению тех же двух действительных значений.
26. Написать и протестировать процедуру для определения целой части и остатка кубического корня из натурального числа x ($\sqrt[3]{x} = e^{\frac{\ln x}{3}}$).
27. Написать и протестировать функцию, которая вычисляет среднее арифметическое, среднее геометрическое и среднее гармоническое двух чисел a_1 и a_2 по следующим формулам: $a_{ар} = \frac{a_1 + a_2}{2}$, $a_{geom} = \sqrt{a_1 \cdot a_2}$, $a_{гарм} = \frac{2}{\frac{1}{a_1} + \frac{1}{a_2}}$.
28. Написать и протестировать функцию нахождения координат (x, y) точки, делящей отрезок AB пополам, если известно, что его концы имеют координаты: $A(x_1, y_1), B(x_2, y_2)$.
29. Даны два действительных числа m и n ($n \neq 0$). Написать и протестировать функцию для определения целой части и остатка от деления m на n .
30. Написать и протестировать функцию нахождения минимума и максимума из 4 действительных значений.

Варианты индивидуальных задач к Заданию 2:

1. Дано натуральное число n ; найти количество его цифр. Использовать программу, включающую рекурсивную функцию.
2. Числа Фибоначчи u_0, u_1, u_2 определяются следующим образом: $u_0 = 0, u_1 = 1, u_n = u_{n-1} + u_{n-2}$ ($n = 2, 3, \dots$). Написать программу вычисления u_n для данного неотрицательного целого n , включающую рекурсивную функцию, которая основана на непосредственном использовании соотношения $u_n = u_{n-1} + u_{n-2}$.

3. При положительном a решением уравнения $x = \frac{2x}{3} + \frac{a}{3x^2}$ служит $x = \sqrt[3]{a}$. Рекуррентное соотношение $x_i = \frac{2x_{i-1}}{3} + \frac{a}{3x_{i-1}^2}$, $x_1 = 1$ можно использовать для быстрого вычисления $\sqrt[3]{a}$, так как элементы последовательности при увеличении i очень быстро приближаются к $\sqrt[3]{a}$. Написать программу вычисления n -ного члена указанной выше последовательности.
4. Написать и протестировать рекурсивную функцию $pow(x, n)$ от вещественного x ($x \neq 0$) и целого n , которая вычисляет величину x^n согласно формуле

$$x^n = \begin{cases} 1 & \text{при } n = 0, \\ 1/x^{|n|} & \text{при } n < 0, \\ x \cdot x^{n-1} & \text{при } n > 0. \end{cases}$$
5. Последовательность чисел задана соотношениями: $a_1 = 2m + n$, $a_i = 2m + \frac{a_{i-1} \cdot n}{a_{i-1} + n}$. Написать и протестировать рекурсивную функцию вычисления n -ного члена этой последовательности.
6. Дано натуральное число n ; найти $n!$. Использовать программу, включающую рекурсивную функцию вычисления $n!$.
7. Написать рекурсивную функцию вычисления суммы $1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$. Обратите для этого $a_i = \frac{x^i}{i!}$ и используйте следующие рекурсивные соотношения:

$$a_i = a_{i-1} \cdot \frac{x}{i}, \quad a_0 = 1, \quad s_i = s_{i-1} + a_i, \quad s_0 = 1.$$
8. Вычислить n -ый член последовательности, заданной следующими соотношениями: $a_1 = 2$, $a_i = 2a_{i-1} - 1$. Использовать программу, включающую рекурсивную функцию.
9. Арифметической прогрессией называется числовая последовательность, каждый член которой, начиная со второго, равен предыдущему, сложенному с некоторым числом. Написать и протестировать рекурсивную функцию вычисления n -ного члена арифметической прогрессии.
10. Написать и протестировать рекурсивную функцию $C(m, n)$, где $0 \leq m \leq n$, для вычисления биномиального коэффициента C_n^m по следующей формуле: $C_n^0 = C_n^n = 1$; $C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$ при $0 < m < n$.
11. Вычислить n -ые члены последовательностей $x_1, x_2, \dots$ и $y_1, y_2, \dots$, если они заданы следующими соотношениями: $x_1 = 1$, $x_i = x_{i-1} + 0.1 \cdot y_{i-1}$; $y_1 = 0$, $y_i = y_{i-1} + 0.1 \cdot x_{i-1}$.

12. Даны натуральные числа n , m , причем $n \geq m$; найти НОД(n , m). Использовать программу, включающую рекурсивную функцию вычисления НОД, основанную на соотношении $\text{НОД}(n, m) = \text{НОД}(m, r)$, где r — остаток от деления n на m .
13. Написать и протестировать рекурсивную функцию для вычисления N -ного члена последовательности, начинающейся с единицы, в которой каждый следующий член равен сумме обратных величин всех предыдущих.
14. Функция $f(n)$ для целых неотрицательных n определена следующим образом:

$$f(n) = \begin{cases} n - 10, & \text{если } n > 100, \\ f(f(n + 11)), & \text{если } n \leq 100. \end{cases}$$
 Вычислить $f(106)$, $f(99)$, $f(85)$. Какие вообще значения принимает эта функция?
15. Написать и протестировать рекурсивную функцию для вычисления N -ного члена последовательности, начинающейся с единицы, в которой каждый следующий член равен сумме квадратов всех предыдущих.
16. Даны натуральные числа a , c , m . Получить $f(m)$, где

$$f(n) = \begin{cases} n, & \text{если } 0 \leq n \leq 9, \\ g(n)f(n - 1 - g(n)), & \text{если } n < 0 \text{ или } n > 9. \end{cases}$$
 Здесь $g(n)$ — остаток от деления $an + c$ на 10. Использовать программу, включающую рекурсивную функцию вычисления $f(n)$.
17. Геометрической прогрессией называется числовая последовательность, каждый член которой, начиная со второго, равен предыдущему, умноженному на некоторое отличное от нуля постоянное число. Написать и протестировать рекурсивную функцию вычисления n -ного члена геометрической прогрессии.
18. Написать и протестировать рекурсивную функцию $\text{root}(f, a, b, \text{eps})$, которая методом деления отрезка пополам находит с точностью eps корень уравнения $f(x)=0$ на отрезке $[a, b]$ (Считать, что $\text{eps} > 0$, $f(a)f(b) < 0$ и $f(x)$ — непрерывная и монотонная функция на отрезке $[a, b]$).
19. Даны неотрицательные целые числа n , m ; вычислить $A(n, m)$, где

$$A(n, m) = \begin{cases} m + 1, & \text{если } n = 0, \\ A(n - 1, 1), & \text{если } n \neq 0, m = 0, \\ A(n - 1, A(n, m - 1)), & \text{если } n > 0, m > 0 \end{cases}$$
 (это — так называемая функция Аккермана). Использовать программу, включающую рекурсивную функцию.
20. Написать и протестировать функцию $\text{min}(x)$ для определения минимального элемента одномерного числового массива x , введя вспомогательную рекурсивную функцию $\text{min}(k)$, находящую минимум среди последних элементов массива x , начиная с k -го.
21. Функция $f(n)$ для целых неотрицательных n определена следующим образом: $f(0) = 0$, $f(1) = 1$, $f(2n) = f(n)$, $f(2n + 1) = f(n) + f(n + 1)$. Для данного N найти и напечатать $f(N)$.
22. Написать и протестировать рекурсивную логическую функцию $\text{symm}(s, i, j)$, проверяющую, является ли симметричной часть строки s , начинающаяся i -м и кончающаяся j -м ее элементами.

23. С клавиатуры вводится последовательность положительных вещественных чисел, за которой следует отрицательное число. Написать и протестировать рекурсивную функцию `sum` без параметров для нахождения суммы этих положительных чисел.
24. Написать и протестировать рекурсивную функцию для вычисления N -го члена последовательности, начинающейся с единицы, в которой каждый следующий член равен синусу суммы всех предыдущих.
25. При положительном a решением уравнения $x = \frac{x}{2} + \frac{a}{2x}$ служит $x = \sqrt{a}$. Рекуррентное соотношение $x_i = \frac{x_{i-1}}{2} + \frac{a}{2x_{i-1}}, x_1 = 1$ можно использовать для быстрого вычисления $\sqrt{a}$, так как элементы последовательности при увеличении i очень быстро приближаются к $\sqrt{a}$. Написать программу вычисления n -го члена указанной выше последовательности.
26. Написать и протестировать рекурсивную функцию для вычисления n -го члена последовательности, заданной следующими соотношениями: $a_i = i \cdot a_{i-1} + a_{i-2}, a_1 = a_2 = 1$.
27. Дано натуральное число n ; найти сумму его цифр. Использовать программу, включающую рекурсивную или функцию.
28. Написать и протестировать рекурсивную процедуру или функцию для вычисления n -го члена последовательности, заданной следующими соотношениями: $a_i = a_{i-1} + a_{i-2} + a_{i-3}, a_1 = a_2 = a_3 = 1$.

Лабораторная работа 12 .

Алгоритмы работы со строками

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие алгоритмы обработки строковых данных с использованием свойств и методов объекта `String`.

Теоретический материал по теме

Строковый тип данных (string) предназначен для обработки строк, т.е. последовательностей символов, заключенных в одинарные или двойные кавычки.

Строковый литерал – это последовательность из нуля или более Unicode-символов, заключенная в одинарные или двойные кавычки.

Примеры строк:

`""` // Это пустая строка: в ней ноль символов

'testing'

"3.14"

'name="myform"'

"Пи - это отношение длины окружности круга к его диаметру" Сами символы двойных кавычек могут содержаться в строках, ограниченных символами одинарных кавычек, а символы одинарных кавычек – в строках, ограниченных символами двойных кавычек. Строковые литералы должны записываться в одной строке программы и не могут разбиваться на две строки.

Программы на JavaScript часто содержат строки HTML-кода, а HTML-код, в свою очередь, часто содержит строки JavaScript-кода. Как и в JavaScript, в HTML для ограничения строк применяются либо одинарные, либо двойные кавычки. Поэтому при объединении JavaScript и HTML-кода есть смысл придерживаться одного «стиля» кавычек для JavaScript, а другого – для HTML.

В следующем примере строка «Спасибо» в JavaScript-выражении заключена в одинарные кавычки, а само выражение, в свою очередь, заключено в двойные кавычки как значение HTML-атрибута обработчика событий:

```
<a href="" onclick="alert('Спасибо')">Щелкни меня</a>
```

Чтобы включить в строковый литерал специальные символы, необходимо использовать нотацию escape-последовательностей:

Код	Вывод	Код	Вывод
\0	нулевой символ (символ NUL)	\v	вертикальная табуляция
\'	одинарная кавычка	\t	табуляция
\"	двойная кавычка	\b	забой
\\	обратный слэш	\f	подача страницы
\n	новая строка	\uXXXX	кодировка Юникода
\r	возврат каретки	\xXX	символ из кодировки Latin-1

Строки полезны для хранения данных, которые можно представить в текстовой форме. Некоторые из наиболее частых операций со строками – это проверка их длины, построение строки с помощью операций строковой

конкатенации + и +=, проверка на существование или местоположение подстрок с помощью метода indexOf(), либо извлечение подстрок с помощью метода substring().

Объект String используется, чтобы представить и конструировать последовательность символов.

Значением **свойства length** объекта **String** является количество символов в стро-ке. Для пустой строки это значение равно нулю.

Синтаксис: **объект.length**

Пример:

```
var txt =  
"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭ  
ЮЯ"; var sln = txt.length; // 33
```

Позиции символов строки нумеруются от нуля до **объект.length-1**.

Стандартные методы объекта String

Метод charAt возвращает символ, находящийся в данной позиции строки. Синтаксис: **объект.charAt(позиция)**

Аргументы: позиция - любое числовое выражение.

Результат: строковое значение.

Метод charAt возвращает строку, состоящую из символа, расположенного в данной позиции примитивного значения строкового объекта. Если позиция лежит вне этого диапазона, то возвращается пустая строка.

Пример:

```
"Строка".charAt(0) // символ С
```

Метод charCodeAt возвращает код символа, находящегося в данной позиции строки.

Синтаксис: **объект.charCodeAt(позиция)**

Аргументы: позиция - любое числовое выражение.

Результат: числовое значение.

Метод `charAt` возвращает число, равную коду Unicode символа, расположенного в данной позиции примитивного значения строкового объекта. Если позиция лежит вне этого диапазона, то возвращается NaN.

Пример:

```
"Строка".charCodeAt(0) // шестнадцатеричный код буквы "С": 421
```

Метод `concat` возвращает конкатенацию строк.

Синтаксис: `объект.concat(строка0, строка1, ..., строкаN)`

Аргументы: `строка0, строка1, ..., строкаN` - любые строковые выражения.

Результат: строковое значение.

Метод `concat` возвращает новую строку, являющуюся конкатенацией исходной строки и аргументов метода. Этот метод эквивалентен операции `объект + строка0 + строка1 + ... + строкаN`

Пример:

```
"Мороз и солнце. ".concat("День чудесный.")
```

Метод `fromCharCode` создает строку из символов, заданных кодами Unicode. Синтаксис: `String.fromCharCode(код1, код2, ..., кодN)` Аргументы: `код1, код2, ..., кодN` - числовые выражения.

Результат: строковое значение.

Метод `fromCharCode` создает новую строку (но не строковый объект), которая является конкатенацией символов Unicode с кодами `код1, код2, ..., кодN`. Это статический метод объекта `String`, поэтому для доступа к нему не нужно специально создавать строковый объект.

Пример:

```
var s = String.fromCharCode(65, 66, 67); // s равно "ABC"
```

Метод `indexOf` возвращает позицию первого вхождения заданной подстроки. Синтаксис: `объект.indexOf(подстрока [,начало]?)`

Аргументы: подстрока - любое строковое выражение, начало - любое числовое выражение.

Результат: числовое значение

Метод `indexOf` возвращает первую позицию подстроки в примитивном значении строкового объекта. Если задан необязательный аргумент `начало`, то поиск ведется, начиная с позиции `начало`; если нет, то с позиции 0, т. е. с первого символа строки. Если `начало` отрицательно, то оно принимается равным нулю; если `начало` больше, чем `объект.length-1`, то оно принимается равным `объект.length-1`. Если объект не содержит данной подстроки, то возвращается значение -1.

Поиск ведется слева направо. В остальном этот метод идентичен методу `lastIndexOf`.

Примеры:

1)

```
var str = "Пожалуйста, найдите, где происходит 'locate'"; var pos =  
str.indexOf("locate");
```

2)

```
//подсчет количества вхождений подстроки pattern в строку str  
function occur(str, pattern) {  
    var pos = str.indexOf(pattern);  
    for (var count = 0; pos != -1; count++)  
        pos = str.indexOf(pattern, pos + pattern.length); return count;  
}
```

Метод `lastIndexOf` возвращает позицию последнего вхождения заданной подстроки.

Синтаксис: `объект.lastIndexOf(подстрока [,начало]?)`

Аргументы: подстрока - любое строковое выражение, начало - любое числовое выражение.

Результат: числовое значение.

Метод `lastIndexOf` возвращает последнюю позицию подстроки в примитивном значении строкового объекта. Если задан необязательный аргумент `начало`, то поиск ведется, начиная с позиции `начало`; если нет, то с позиции 0, т. е. с первого символа строки. Если `начало` отрицательно, то оно принимается равным нулю; если `начало` больше, чем `объект.length-1`, то оно принимается равным `объект.length-1`. Если объект не содержит данной подстроки, то возвращается значение -1.

Поиск ведется справа налево. В остальном этот метод идентичен методу `indexOf`.

Примеры:

1)

```
var n = "Белый кит".lastIndexOf("кит"); // n равно 6
```

2)

```
var str = "Пожалуйста, найдите, где происходит 'locate'"; var pos =  
str.lastIndexOf("locate");
```

Метод `localeCompare` сравнивает две строки с учетом языка операционной системы.

Синтаксис: `объект.localeCompare(строка1)`

Аргументы: `строка1` - любое строковое выражение

Результат: число

Метод `localeCompare` сравнивает две строки с учетом национальных установок операционной системы. Он возвращает -1, если примитивное значение объекта меньше `строка1`, +1, если оно больше `строка1`, и 0, если эти значения совпадают.

Метод match сопоставляет строку с регулярным выражением.

Синтаксис: объект.match(регвыр)

Аргументы: регвыр - любое регулярное выражение.

Результат: массив строк.

Метод match сопоставляет регулярное выражение регвыр с примитивным значени-ем строкового объекта. Результатом сопоставления является массив найденных подстрок или null, если соответствий нет. При этом:

- Если регвыр не содержит опцию глобального поиска, то выполняется метод регвыр.exec(объект) и возвращается его результат. Результирующий массив содержит в элементе с индексом 0 найденную подстроку, а в остальных элементах — подстроки, соответствующие подвыражениям регвыр, заключенным в круглые скобки.
- Если регвыр содержит опцию глобального поиска, то метод регвыр.exec(объект) выполняется до тех пор, пока находятся соответствия. Если n — количество найденных соответствий, то результатом является массив из n элементов, которые содержат найденные подстроки. Свойству регвыр.lastIndex присваивается номер позиции в исходной строке, указывающий на первый символ после последнего найденного соответствия, или 0, если соответствий не найдено.

Следует помнить, что метод регвыр.exec изменяет свойства объекта регвыр.

Примеры:

```
var src = "Он сказал: <I>Я ухожу</I> и добавил: <I>До свидания</I>."; var  
res = src.match(/<i>.*?<\i>/i); // Я ухожу
```

```
var res = src.match(/<i>.*?<\i>/ig); // Я ухожу, До свидания
```

Метод replace сопоставляет строку с регулярным выражением и заменяет найденную подстроку новой подстрокой.

Синтаксис: объект.replace(регвыр,строка)

объект.replace(регвыр,функция)

Аргументы: регвыр - регулярное выражение, строка - строковое выражение, функция - имя функции или декларация функции.

Результат: новая строка.

Метод `replace` сопоставляет регулярное выражение регвыр с примитивным значением строкового объекта и заменяет найденные подстроки другими подстроками. Результатом является новая строка, которая является копией исходной строки с проведенными заменами. Способ замены определяется опцией глобального поиска в регвыр и типом второго аргумента.

- Если регвыр не содержит опцию глобального поиска, то выполняется поиск первой подстроки, соответствующей регвыр и производится ее замена. Если регвыр содержит опцию глобального поиска, то выполняется поиск всех подстрок, соответствующих регвыр, и производится их замена.
- Если вторым аргументом является строка, то замена каждой найденной подстроки производится на нее. При этом строка может содержать такие свойства объекта

`RegExp`, как `$1`, ..., `$9`, `lastMatch`, `lastParen`, `leftContext` и `rightContext`.

Пример:

```
"Вкусные яблоки, сочные яблоки.".replace(/яблоки/g, "груши"))  
// Вкусные груши, сочные груши.
```

Если вторым аргументом является функция, то замена каждой найденной подстроки производится вызовом этой функции. Функция имеет следующие аргументы. Первый аргумент - это найденная подстрока, затем следуют аргументы, соответствующие всем подвыражениям регвыр, заключенным в круглые скобки, предпоследний аргумент - это позиция найденной подстроки в исходной строке, считая с нуля, и последний аргумент - это сама исходная строка.

Пример, показывающий, как с помощью метода `replace` можно написать функцию преобразования градусов Фаренгейта в градусы Цельсия:

```
function myfunc($0,$1) {
    return (($1-32) * 5 / 9) + "C";
}
function f2c(x) {
    var s = String(x);
    return s.replace(/(\d+(\.\d*)?)F\b/, myfunc);
}
document.write(f2c("212F")); // выведет строку 100C
```

Следует помнить, что этот метод изменяет свойства объекта регвыр.

Пример использования replace для замены всех вхождений подстроки в строку:

```
var str = "foobarfoobar";
str=str.replace(/foo/g,"xxx"); // str = "xxxbarxxxbar"
```

Метод search ищет сопоставление строки с регулярным выражением.

Синтаксис: объект.search(регвыр)

Аргументы: регвыр - любое регулярное выражение.

Результат: числовое выражение.

Метод search сопоставляет регулярное выражение регвыр с примитивным значением строкового объекта. Результатом сопоставления является позиция первой найденной подстроки, считая с нуля, или -1, если соответствий нет. При этом опция глобального поиска в регвыр игнорируется, и свойства регвыр не изменяются.

Примеры:

```
var src = "Он сказал: <I>Я ухожу</I> и добавил: <I>До свидания</I>."; var
n = src.search(/<i>.*?<\i>/i); // n равно 11
```

Метод slice извлекает часть строки и возвращает новую строку.

Синтаксис: объект.slice(начало [,конец]?)

Аргументы: начало и конец - любые числовые выражения.

Результат: новая строка.

Метод `slice` возвращает подстроку примитивного значения строкового объекта, от позиции начало до позиции конец, не включая ее. Если конец не задан, то возвращается подстрока, начиная с позиции начало и до конца исходной строки.

Если значение начало отрицательно, то оно заменяется на `объект.length+начало`. Если значение конец отрицательно, то оно заменяется на `объект.length+конец`. Иными словами, отрицательные аргументы трактуются как смещения от конца строки.

Примеры:

1)

```
var str = "Яблоко, Банан, Киви";  
var res = str.slice(7, 13); // "Банан"
```

2)

```
document.write("ABCDEF".slice(2,-1))           // выведет строку CDE.
```

Метод `split` разбивает строку на массив подстрок. Синтаксис:

`объект.split(разделитель [,число]?)`

Аргументы: разделитель - строковое или регулярное выражение, число - числовое выражение.

Результат: массив строк (объект `Array`).

Метод `split` разбивает примитивное значение объекта на массив подстрок и возвращает его. Разбиение на подстроки производится следующим образом. Исходная строка просматривается слева направо в поисках разделителя. Как только он найден, подстрока от конца предыдущего разделителя (или от начала строки, если это первое вхождение разделителя) до начала найденного добавляется в массив подстрок. Таким образом, сам разделитель в текст подстроки не попадает.

Необязательный аргумент число задает максимально возможный размер результирующего массива. Если он задан, то после выделения числа

подстрока метод завершает работу, даже если просмотр исходной строки не закончен.

Разделитель может быть задан либо строкой, либо регулярным выражением. Существует несколько случаев, требующих особого рассмотрения:

- Если разделитель не задан, то результирующий массив состоит из одного элемента, равного исходной строке.
- Если разделитель — пустая строка или регулярное выражение, соответствующее пустой строке, то результирующий массив состоит из `объект.length` элементов, каждый из которых содержит один символ исходной строки.
- Если разделитель — регулярное выражение, то очередная подстрока выделяется всякий раз, когда нашлось соответствие регулярному выражению, и соответствующая подстрока служит разделителем.

Примеры:

1)

```
var s="a1b2c3d".split(/d/); // массив ["a", "b", "c", "d"]
```

2) регулярное выражение используется для задания тегов HTML в качестве

разделителя `document.write("Текст <B>полужирный</B> и`

`<EM>курсивный</EM>".split(/<\/?([<>]+)>/));`

`// выведет строку Текст ,полужирный, и ,курсивный.`

Метод `substr` возвращает подстроку, заданную позицией

и длиной. Синтаксис: `объект.substr(позиция [,длина]?)`

Аргументы: позиция и длина - числовые выражения.

Результат: строковое значение

Метод `substr` возвращает подстроку примитивного значения строкового объекта, начинающуюся с данной позиции и содержащую длина символов. Если длина не задана, то возвращается подстрока, начиная с данной позиции

и до конца исходной строки. Если длина отрицательна или равна нулю, то возвращается пустая строка.

Если позиция больше или равна `объект.length`, то возвращается пустая строка. Если позиция отрицательна, то она трактуется как смещение от конца строки, т. е. заменяется на `объект.length+позиция`.

Примеры:

```
var src = "abcdef";  
var s1   = src.substr(1, 3);           // "bcd"  
var s2   = src.substr(1);              // "bcdef"  
var s3   = src.substr(-1);             // "f"
```

Метод substring возвращает подстроку, заданную начальной и конечной позициями.

Синтаксис: `объект.substring(начало [,конец])`

Аргументы: начало и конец - числовые выражения.

Результат: строковое значение

Метод `substring` возвращает подстроку примитивного значения строкового объекта, от позиции `начало` до позиции `конец`, не включая ее. Если `конец` не задан, то возвращается подстрока, начиная с позиции `начало` и до конца исходной строки.

Отрицательные аргументы или равные `NaN` заменяются на нуль; если аргумент больше длины исходной строки, то он заменяется на нее. Если `начало` больше `конца`, то они меняются местами. Если `начало` равно `концу`, то возвращается пустая строка.

Примеры:

1)

```
var src = "abcdef";  
var s1 = src.substring(1, 3);           // "bc"  
var s2  = src.substring(1, -1);         // "a"  
var s3  = src.substring(-1, 1);         // "a"
```

2)

```
var str = "Яблоко, Банан,      Киви";  
var res = str.substring(7, 13);           // "Банан"
```

Метод toLowerCase преобразует все буквы строки в строчные.

Синтаксис: объект.toLowerCase()

Результат: новая строка.

Метод toLowerCase возвращает новую строку, в которой все буквы исходной строки заменены на строчные. Остальные символы исходной строки не изменяются. Исходная строка остается прежней.

Пример:

```
document.write("объект String".toLowerCase()) // объект string
```

Метод toUpperCase преобразует все буквы строки в прописные.

Синтаксис: объект.toUpperCase()

Результат: новая строка

Метод toUpperCase возвращает новую строку, в которой все буквы исходной строки заменены на прописные. Остальные символы исходной строки не изменяются. Исходная строка остается прежней.

Пример:

```
document.write("объект String".toUpperCase()) // ОБЪЕКТ STRING
```

Метод toString преобразует объект в строку.

Синтаксис: объект.toString()

Результат: строковое значение

Метод toString возвращает примитивное значение строкового объекта.

Примеры:

```
var number = -507;  
string_value = number.toString(); var n = 17;  
binary_string = n.toString(2); // Вернет "10001"
```

```
octal_string = "0" + n.toString(8); // Вернет "021"  
hex_string = "0x" + n.toString(16); // Вернет "0x11"
```

Метод valueOf возвращает примитивное значение объекта.

Синтаксис: объект.valueOf()

Результат: строковое значение

Метод valueOf возвращает примитивное значение строкового объекта.

Пример:

```
var x = new String('Привет, мир');  
document.write (x.valueOf()); // Отобразит 'Привет, мир'
```

Задания для выполнения

Задание 1. Разработайте скрипт, реализующий алгоритм обработки строковых данных. Ввод исходных данных осуществлять с использованием функции prompt. Вывод полученных результатов осуществлять с использованием функции alert. Сохраните скрипт в файле **script_4.js**.

Задание 2. Создайте по образцу html-документ **lab_4.html** и подключите к нему внешний скрипт **script_4.js**. Для тестирования работоспособности скрипта разработайте 3 теста (включающих наборы входных и выходных данных). Выполните тестирование и отладку скрипта в браузере.

Варианты индивидуальных задач к Заданию 1:

1. Даны две строки s_1 и s_2 . Посчитать, сколько раз в строке s_2 встречается первый символ строки s_1 .
2. Дана строка s , состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Найти номер самого длинного слова. Если таких слов несколько, то указать все их номера.
3. Дана строка s . Известно, что среди символов данной строки имеется хотя бы один пробел и что первый символ строки s отличен от пробела. Преобразовать строку s , заменив все малые буквы латинского алфавита, предшествующие первому пробелу, одноименными большими.
4. Даны две строки s_1 и s_2 , имеющие одинаковую длину. Образовать строку s , в которой должны чередоваться символы строк s_1 и s_2 .
5. Дана строка s , состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Если количество слов в строке s четно, то поменять местами ее средние слова. В противном случае на место среднего слова строки s поместить пробелы.
6. Дана строка s . Преобразовать ее, заменив в ней каждую точку многоточием (то есть тремя точками), а все восклицательные знаки вопросительными.
7. Дана строка s . Преобразовать эту строку так, чтобы ее символы шли в обратном по-рядке по отношению к их исходному расположению.
8. Дана строка s , состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, сколько слов состоит из одинаковых символов, например: ааааа, еее.
9. Определить долю пробелов в данной строке s .

10. Даны две строки s_1 и s_2 , имеющие различную длину. Посчитать в них количество попарно одинаковых символов.
11. Дана строка s , состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить число слов, в которых последняя буква в слове не повторяется.
12. Дана строка s . Определить, является ли она "палиндромом". Если нет, то выяснить, станет ли строка s "палиндромом" после удаления из нее всех пробелов.
13. Дана строка s_1 . Образовать строку s_2 , включив в нее символы строки s_1 , расположенные на нечетных позициях, кроме пробелов и знаков препинания.
14. Дана строка s , состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, повторяется ли в тексте первое слово.
15. Дана строка s . Вывести ее на экран, подчеркивая (ставя минусы в соответствующих позициях следующей строки) все входящие в него цифры.
16. Дана строка s . Исключить из нее все лишние пробелы, то есть вместо каждой группы пробелов оставить только один. При исключении пробелов текст сдвигается влево.
17. Дана строка s , состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Найти номер самого короткого слова. Если таких слов несколько, то указать все их номера.

18. Даны две строки s_1 и s_2 , имеющие различную длину. Посчитать в них количество попарно различных символов.
19. Дана строка s , состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, сколько раз стоящие рядом два слова начинаются на одну и ту же букву.
20. Дана строка s . Определить в ней долю символов, являющихся буквами латинского алфавита.
21. Дана строка s , состоящая из букв латинского алфавита. Верно ли, что в данной строке гласные буквы (a, e, i, o, u) чередуются с согласными?
22. Дана строка s . Известно, что среди символов данной строки есть по крайней мере один восклицательный знак и что первый символ этой строки отличен от восклицательного знака. Выяснить, имеется ли среди символов строки s , предшествующих первому восклицательному знаку, пара соседствующих одинаковых символов.
23. Дана строка s , состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Если первое и последнее слова строки имеют одинаковую длину, то поменять их местами. В противном случае оставить строку s без изменения.
24. Дана строка s . Удалить из нее все группы букв вида $abcd$.
25. Дана строка s , состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить число слов в строке, состоящих из нечетного числа символов.
26. Дана строка s , состоящая из букв латинского алфавита. Преобразовать данную строку, упорядочив в ней буквы по алфавиту.

27. Дана строка s , состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, содержит ли данная строка слова, заканчивающиеся символом $*$.
28. Дана строка s . Известно, что среди символов данной строки имеется хотя бы один пробел и что первый символ строки s отличен от пробела. Преобразовать строку s , удалив из нее все символы, предшествующие первому пробелу и не являющиеся буквами латинского алфавита.
29. Дана строка s . Выяснить, имеются ли такие натуральные i и j , что $i < j$ и i -тый символ строки s совпадает с $(i+1)$ -ым символом, а j -тый — с $(j+1)$ -ым символом строки s .
30. Дана строка s , состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Преобразовать строку s , удалив из нее все повторные вхождения слов.

Лабораторная работа 13.

Массивы с числовыми индексами

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие алгоритмы обработки одномерных и многомерных (двумерных) числовых массивов.

Теоретический материал по теме

Массив – тип данных, содержащий (хранящий) пронумерованные значения. Каждое пронумерованное значение называется **элементом массива**, а число, с которым связывается элемент, называется его **индексом**.

Так как JavaScript – нетипизированный язык, элемент массива может иметь любой тип, причем разные элементы одного массива могут иметь разные типы. Элементы массива могут даже содержать другие массивы, что позволяет создавать массивы массивов. Индексация элементов массива начинается с 0.

Массивы в JavaScript являются динамическими: они могут увеличиваться и уменьшаться в размерах по мере необходимости; нет необходимости объявлять фиксированные размеры массивов при их создании или повторно распределять память при изменении их размеров.

1. Создание массивов

Создать массив с помощью **литерала**, который представляет собой простой список разделенных запятыми элементов массива в квадратных скобках. Значения в литерале массива не обязательно должны быть константами - это могут быть любые выражения, в том числе и литералы объектов.

Примеры:

```
let empty = []; // Пустой массив
let primes = [2, 3, 5, 7, 11]; // Массив с 5 числовыми элементами
let misc = [1.1, true, "a"]; // 3 элемента разных типов
let base = 1024;
let table = [base, base+1, base+2, base+3]; let b = [[1, {x:1, y:2}], [2, {x:3, y:4}]];
let count = [1,,3]; // Массив из 3 элементов, средний элемент не определен.
let undefs = [,,]; // Массив из 2 элементов, оба не определены.
```

Другой способ создания массива состоит в вызове конструктора `Array()`.

Вызвать конструктор можно тремя разными способами:

1) Вызвать конструктор без аргументов:

```
let arr = new Array();
```

В этом случае будет создан пустой массив, эквивалентный литералу `[]`.

2) Вызвать конструктор с единственным числовым аргументом, определяющим длину массива:

```
let arr = new Array(10);
```

В этом случае будет создан пустой массив указанной длины. Такая форма вызова конструктора `Array()` может использоваться для предварительного

распределения памяти под массив, если заранее известно количество его элементов. При этом в массиве не сохраняется никаких значений.

3) Явно указать в вызове конструктора значения первых двух или более элементов массива или один нечисловой элемент:

```
let arr = new Array(5, 4, 3, 2, 1, "тест");
```

В этом случае аргументы конструктора становятся значениями элементов нового массива. Примеры:

```
let a = new Array( ); // пустой массив
```

```
let a = new Array(5, 4, 3, 2, 1, "testing, testing"); let a = new Array(10);
```

```
// массив из 10 элементов
```

2. Чтение и запись элементов массива

Доступ к элементам массива осуществляется с помощью оператора []. Слева от скобок должна присутствовать ссылка на массив. Внутри скобок должно находиться произвольное выражение, возвращающее неотрицательное целое значение. Этот синтаксис пригоден как для чтения, так и для записи значения элемента массива.

Примеры:

```
value = a[0]; a[1] =
```

```
3.14;
```

```
    = 2;
```

```
a[i] = 3;
```

```
a[i + 1] = "hello";
```

```
a[a[i]] = a[0];
```

```
my['salary'] *= 2;
```

```
a[1.23] = true; //создается новое свойство с именем "1.23"
```

3. Длина массива

Все массивы имеют свойство `length`, устанавливающее количество элементов в массиве. Поскольку массивы могут иметь неопределенные элементы, свойство `length` всегда на единицу больше, чем самый большой номер элемента массива.

Примеры:

```
let a = new Array(); //a.length == 0 (ни один элемент не определен)
a = new Array(10); //a.length == 10 (определены пустые элементы 0–9)
a = new Array(1,2,3); //a.length == 3 (определены элементы 0–2)
a = [4, 5]; // a.length == 2 (определены элементы 0 и 1)
a[5] = 1; // a.length == 6 (определены элементы 0, 1 и 5)
a[49] = 0; // a.length == 50 (определены элементы 0, 1, 5 и 49)
```

Свойство `length` массива доступно как для чтения, так и для записи. Если установить свойство `length` в значение, меньшее текущего, массив укорачивается до новой длины; любые элементы, не попадающие в новый диапазон индексов, отбрасываются, и их значения теряются. Если сделать свойство `length` большим, чем его текущее значение, в конец массива добавляются новые неопределенные элементы, увеличивая массив до нового размера.

4. Обход элементов массива

Одним из самых старых способов перебора элементов массива является цикл `for` по цифровым индексам.

Примеры:

1)

```
let fruits = ["манго", "банан", "вишня", "персик"]; for(let i = 0; i <
fruits.length; i++)
    alert(fruits[i]);
```

2)

```
let lookup_table = new Array(1024);
for(let i = 0; i < lookup_table.length; i++)
    lookup_table[i] = i * 512;
```

3)

```
let k = Number(prompt('Введите размер массива ', '5')); let x = new
Array(k);
for (let i=0; i<=k-1; i++)
    x[i] = Math.round((Math.random()*100));
```

Также для массивов возможен и другой вариант цикла, `for..of`. Цикл `for..of` не предоставляет доступа к номеру текущего элемента, только к его значению.

Пример:

```
let fruits = ["Яблоко", "Апельсин", "Слива"];  
//  проходит по значениям
```

```
for (let fruit of fruits) { alert( fruit );  
}
```

Методы класса Array

Метод `join()`

Метод `Array.join()` преобразует все элементы массива в строки, объединяет их и возвращает получившуюся строку. В необязательном аргументе методу можно передать строку, которая будет использоваться для отделения элементов в строке результата. Если строка-разделитель не указана, используется запятая.

Пример:

```
let arr = [1,2,3];  
arr.join();                // '1,2,3'  
arr.join("-");             // '1-2-3'
```

Метод `reverse()`

Метод `Array.reverse()` меняет порядок следования элементов в массиве на обратный и возвращает переупорядоченный массив. Перестановка выполняется непосредственно в исходном массиве, т.е. этот метод не создает новый массив с переупорядоченными элементами, а переупорядочивает их в уже существующем массиве.

Пример:

```
let arr = [1,2,3];  
arr.reverse().join();      // "3,2,1"
```

Метод `sort()`

Метод `Array.sort()` сортирует элементы в исходном массиве и возвращает отсортированный массив. Если метод `sort()` вызывается без аргументов, сортировка выполняется в алфавитном порядке (для сравнения элементы временно преобразуются в строки, если это необходимо). Неопределенные элементы переносятся в конец массива.

Для сортировки в каком-либо ином порядке, отличном от алфавитного, методу `sort()` можно передать функцию сравнения в качестве аргумента. Эта функция устанавливает, какой из двух ее аргументов должен следовать раньше в отсортированном списке.

Если первый аргумент должен предшествовать второму, функция сравнения должна возвращать отрицательное число. Если первый аргумент должен следовать за вторым в отсортированном массиве, то функция должна возвращать число больше нуля. А если два значения эквивалентны (т.е. порядок их следования не важен), функция сравнения должна возвращать 0.

Пример:

```
let arr = [33, 4, 1111, 222];
arr.sort(); // Алфавитный порядок: 1111, 222, 33, 4
arr.sort(function(a,b) {

    // Числовой порядок: 4, 33, 222, 1111
    return a-b; // Возвращает значение < 0, 0 или > 0
                // в зависимости от порядка сортировки a и b
});
// Сортируем в обратном направлении, от большего к меньшему
arr.sort(function(a,b) {return b-a});
```

Метод `concat()`

Метод `Array.concat()` создает и возвращает новый массив, содержащий элементы исходного массива, для которого был вызван метод `concat()`, и значения всех аргументов, переданных методу `concat()`. Если какой-либо из этих аргументов сам является массивом, его элементы добавляются в возвращаемый

массив. Следует, однако, отметить, что рекурсивного превращения массива из массивов в одномерный массив не происходит. Метод `concat()` не изменяет исходный массив.

Примеры:

```
let arr = [1,2,3];  
arr.concat(4, 5);           // Вернет [1,2,3,4,5]  
arr.concat([4,5]);          // Вернет [1,2,3,4,5]  
arr.concat([4,5],[6,7])     // Вернет [1,2,3,4,5,6,7]  
arr.concat(4, [5,[6,7]])    // Вернет [1,2,3,4,5,[6,7]]
```

Метод `slice()`

Метод `Array.slice()` возвращает фрагмент, или подмассив, указанного массива. Два аргумента метода определяют начало и конец возвращаемого фрагмента. Возвращаемый массив содержит элемент, номер которого указан в первом аргументе, плюс все последующие элементы, вплоть до (но не включая) элемента, номер которого указан во вто-ром аргументе.

Если указан только один аргумент, возвращаемый массив содержит все элементы от начальной позиции до конца массива. Если какой-либо из аргументов имеет отрицательное значение, он определяет номер элемента относительно конца массива. Так, аргументу `-1` соответствует последний элемент массива, а аргументу `-3` - третий элемент массива с конца.

Примеры:

```
let arr = [1,2,3,4,5];  
arr.slice(0,3);             // Вернет [1,2,3]  
arr.slice(3);               // Вернет [4,5]  
arr.slice(1,-1);            // Вернет [2,3,4]  
arr.slice(-3,-2);           // Вернет [3]
```

Метод `splice()`

Метод `Array.splice()` - это универсальный метод, выполняющий вставку или удаление элементов массива. В отличие от методов `slice()` и `concat()`, метод `splice()` изменяет исходный массив, относительно которого он был вызван. Обратите внимание, что методы `splice()` и `slice()` имеют очень похожие имена, но выполняют совершенно разные операции.

Метод `splice()` может удалять элементы из массива, вставлять новые элементы или выполнять обе операции одновременно. Элементы массива при необходимости смещаются, чтобы после вставки или удаления образовывалась непрерывная последовательность.

Первый аргумент метода `splice()` определяет позицию в массиве, начиная с которой будет выполняться вставка и/или удаление. Второй аргумент определяет количество элементов, которые должны быть удалены (вырезаны) из массива. Если второй аргумент опущен, удаляются все элементы массива от указанного до конца массива. Метод `splice()` возвращает массив удаленных элементов или (если ни один из элементов не был удален) пустой массив.

Первые два аргумента метода `splice()` определяют элементы массива, подлежащие удалению. За этими аргументами может следовать любое количество дополнительных аргументов, определяющих элементы, которые будут вставлены в массив, начиная с позиции, указанной в первом аргументе

Примеры:

```
let arr = [1,2,3,4,5,6,7,8];  
arr.splice(4);           // Вернет [5,6,7,8], arr = [1,2,3,4]  
arr.splice(1,2);         // Вернет [2,3], arr = [1,4]  
arr.splice(1,1);         // Вернет [4]; arr = [1]  
arr = [1,2,3,4,5];  
arr.splice(2,0,'a','b'); // Вернет []; arr = [1,2,'a','b',3,4,5]
```

Методы `push()` и `pop()`

Методы `push()` и `pop()` позволяют работать с массивами как со стеками. Метод `push()` добавляет один или несколько новых элементов в конец массива и возвращает его новую длину. Метод `pop()` выполняет обратную операцию -

удаляет последний элемент массива, уменьшает длину массива и возвращает удаленное им значение. Обратите внимание, что оба эти метода изменяют исходный массив, а не создают его модифицированную копию.

Примеры:

```
var arr = []; // Создать пустой массив arr.push('zero'); //
```

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

Методы unshift() и shift()

Методы unshift() и shift() ведут себя почти так же, как push() и pop(), за исключением того, что они вставляют и удаляют элементы в начале массива, а не в конце. Метод unshift() смещает существующие элементы в сторону больших индексов для освобождения места, добавляет элемент или элементы в начало массива и возвращает новую длину массива. Метод shift() удаляет и возвращает первый элемент массива, смещая все последующие элементы на одну позицию вниз, чтобы занять место, освободившееся в начале массива.

Многомерные массивы

Согласно практике решения множества задач и в случае работы со сложными наборами данных возникает необходимость построения многомерных массивов данных. Фактически, Javascript не поддерживает подобных массивов, но синтаксис языка подразумевает использование в качестве элементов массива любого объекта, в том числе и других массивов.

Пример:

```
// Создать многомерный массив
```

```
let table = new Array(10); // В таблице 10 строк
```

```
for(let i = 0; i < table.length; i++)
```

```
    table[i] = new Array(10); // В каждой строке 10 столбцов
```

```
// Заполнить массив значениями и вывести в консоль for(let row = 0,
```

```
str = "; row < table.length; row++) {
```

```
    for(let col = 0; col < table[row].length; col++) { table[row][col] =
```

```
        (row+1)*(col+1);
```

```

        str += table[row][col] + ' ';
    }
    console.log(str + '\n'); str = "";
}

```

Задания для выполнения

Задание 1. Разработайте скрипты, реализующие алгоритм обработки одномерного числового массива. Ввод исходных данных в массив осуществлять двумя способами (на выбор): 1) с использованием функции `prompt`; 2) с использованием функции `Math.random()`, генерирующей псевдослучайные числа. Вывод полученных результатов осуществлять двумя способами (на выбор): 1) с использованием функции `alert`; 2) в ячейки таблицы html-документа. Сохраните скрипт в файле **script_5_1.js**.

Задание 2. Разработайте скрипты, реализующий алгоритм обработки двумерного числового массива. Ввод исходных данных в массив осуществлять двумя способами (на выбор): 1) с использованием функции `prompt`; 2) с использованием функции `Math.random()`, генерирующей псевдослучайные числа. Вывод полученных результатов осуществлять двумя способами (на выбор): 1) с использованием функции `alert`; 2) в ячейки таблицы html-документа. Сохраните скрипт в файле **script_5_2.js**.

Задание 3. Создайте по образцу html-документы **lab_5_1.html** и **lab_5_2.html** и подключите к ним внешние скрипты **script_5_1.js** и **script_5_2.js** соответственно. Для тестирования работоспособности каждого скрипта разработайте по 3 теста (включающих наборы входных и выходных данных). Выполните тестирование и отладку скриптов в браузере.

Варианты индивидуальных задач к заданию 1:

1. Дан одномерный числовой массив. Если он упорядочен по убыванию, то вывести его на экран в обратном порядке; в противном случае вывести на экран номер первого элемента, нарушающего упорядоченность.

2. Дан одномерный числовой массив. Все элементы массива, кратные 3, домножить на их индекс, а затем найти среднее арифметическое элементов преобразованного массива.
3. В одномерном числовом массиве поменять местами элементы с четными и нечетными индексами, если оба они кратны минимальному, и заменить эти элементы их индексами в обратном случае.
4. Дан одномерный числовой массив, все элементы которого различны. Если сумма всех элементов, расположенных между максимальным и минимальным элементами массива, меньше 10, то вычислить произведение ненулевых элементов одномерного числового массива; в противном случае найти количество элементов массива, кратных минимальному, и их произведение.
5. Дан одномерный числовой массив, все элементы которого различны. Подсчитать в нем количество четных элементов, расположенных между наибольшим и наименьшим элементами массива.
6. Дан одномерный числовой массив, все элементы которого различны. Найти наименьшее положительное значение элемента массива и его номер. Если этот номер окажется больше 5, то заменить все элементы, расположенные между наибольшим и наименьшим элементами массива, их квадратами, иначе — кубами.
7. Дан одномерный числовой массив, все элементы которого различны. Посчитать в нем количество соседств из двух положительных чисел. Если это количество кратно 3, то вывести на экран все элементы расположенные до максимального, иначе — после максимального.
8. Дан одномерный числовой массив, все элементы которого различны. Посчитать сумму всех неотрицательных элементов массива с четными индексами. Если эта сумма окажется больше 7, то заменить наибольший и наименьший элементы массива их произведением, иначе — поменять их местами.

9. Дан упорядоченный массив $a_1, a_2, \dots, a_n$. Известно, что число x принадлежит отрезку числовой оси, вмещающему заданный массив. Определить номер k , для которого $a_{k-1} \leq x \leq a_k$. Если k окажется кратным 3, то вывести на экран элементы массива в обратном порядке.
10. Дан одномерный числовой массив, все элементы которого различны. Посчитать сумму кубов элементов с нечетными индексами, расположенных до наибольшего элемента массива, и произведение индексов всех элементов, которые по числовому значению превосходят минимальный, но не превосходят максимальный элемент массива.
11. Дан одномерный числовой массив. Определить, сколько в нем имеется пар совпадающих по величине соседних чисел; вывести на экран их индексы.
12. Дан одномерный числовой массив. Добавить к каждому элементу массива, кроме первого, значение предыдущего. После такого преобразования вычислить среднее арифметическое наибольшего и наименьшего элементов массива.
13. Найти наибольший и наименьший элементы одномерного числового массива, а также подсчитать количество элементов, кратных наименьшему элементу массива.
14. Дан одномерный числовой массив, все элементы которого различны. Заменить в нем четные элементы, расположенные после максимального, их квадратами, а нечетные элементы, расположенные после максимального — их кубами.
15. Вычислить произведение всех ненулевых элементов одномерного числового массива с нечетными индексами. Заменить все элементы массива, равные нулю, отношением их индекса к значению максимального элемента.
16. Заменить в одномерном числовом массиве элементы, большие числа M , на число a , введенное с клавиатуры, а меньшие M — отношением i , где i — индекс элемента.

17. В одномерном числовом массиве заменить отрицательные элементы их квадратами, а положительные разделить на числовое значение минимального элемента.
18. Дан одномерный числовой массив. Определить количество элементов этого массива, которые имеют четные индексы и являются кратными 3, а также вычислить сумму и произведение элементов, кратных либо первому, либо последнему элементам массива.
19. Определить в одномерном числовом массиве число соседств из двух чисел, сумма которых равна 5. Вычислить произведение всех элементов, удовлетворяющих этому условию.
20. Найти суммы и произведения отрицательных и положительных элементов одномерного числового массива. Подсчитать, сколько имеется в массиве элементов, равных нулю.
21. Дан одномерный числовой массив. Домножить на 3 все его положительные элементы с нечетными индексами; все отрицательные элементы, имеющие четные индексы, уменьшить в 5 раз.
22. Дан одномерный числовой массив. Вычислить сумму квадратов его элементов с четными индексами, а также произведение всех ненулевых элементов с нечетными индексами.
23. Определить в одномерном числовом массиве количество соседств из взаимно противоположных и не взаимно обратных чисел.
24. Добавить к каждому нечетному элементу одномерного целочисленного массива его номер, а затем все четные элементы заменить произведением номера элемента на отношение наибольшего элемента массива к наименьшему.
25. Найти наибольший элемент одномерного числового массива из стоящих на нечетных местах и наименьший из стоящих на четных местах. Если сумма этих элементов больше значения первого элемента массива, то уменьшить их в 2 раза, иначе — увеличить каждый из них на 3.

26. Дан одномерный числовой массив, все элементы которого различны. Посчитать, сколько в нем содержится элементов, совпадающих по абсолютной величине с номером. Если таких элементов нет, то вычислить сумму и произведение наибольшего и наименьшего элементов массива.
27. В одномерном числовом массиве найти количество и сумму элементов, кратных 3, а также среднее арифметическое всех элементов, имеющих четные индексы и кратных
28. Определить в одномерном числовом массиве количество соседств из чисел разного знака. Вывести на экран последовательность из чисел одного знака, имеющую наибольшую длину.
29. Дан одномерный числовой массив. Найти сумму корней квадратных из абсолютных величин его элементов, а также произведение всех отрицательных элементов с четными номерами и произведение всех положительных элементов с нечетными номерами.
30. Посчитать в одномерном числовом массиве количество элементов, равных первому отрицательному. Вывести на экран самую длинную подпоследовательность из элементов, равных нулю.

Варианты индивидуальных задач к Заданию 2:

2. Дана действительная квадратная матрица порядка n (n — натуральное число). Заменить нулями все элементы матрицы, находящиеся на ее главной и побочной диагоналях, кроме наибольшего и наименьшего из них.
3. Дана действительная матрица размера $M \times N$. Найти сумму наибольших значений элементов ее строк, а также произведение наименьших элементов ее столбцов.
4. Дано целое число n . Построить целочисленную квадратную матрицу порядка n вида

$$\begin{array}{ccccccc}
 & & n & & n & & \\
 n & 1 & & 2 & & \dots & 1 \\
 & & n & & & & \\
 0 & n & & 1 & & \dots & 2
 \end{array}$$

$$\begin{pmatrix}
 0 & 0 & n & \dots & 3 \\
 \dots & \dots & \dots & \dots & \dots \\
 0 & 0 & 0 & \dots & n
 \end{pmatrix}$$

5. Дана целочисленная квадратная матрица порядка 8. Найти наименьшее из значений элементов столбца, который обладает наибольшей суммой модулей элементов. Если таких столбцов несколько, то взять первый из них.
6. Дана действительная квадратная матрица порядка n , в которой не все элементы равны нулю. Получить новую матрицу путем деления всех элементов данной матрицы, лежащих ниже главной диагонали, на ее наибольший по модулю элемент.
7. Дана действительная квадратная матрица порядка n . Вычислить сумму тех из ее элементов, расположенных на главной диагонали и выше нее, которые превосходят по величине все элементы, расположенные ниже главной диагонали. Если на главной диагонали и выше нее нет элементов с указанным свойством, то вывести сообщение об этом.
8. Дана целочисленная квадратная матрица порядка n . Заменить нулями все неотрицательные элементы этой матрицы, находящиеся на ее побочной диагонали.
9. Дана действительная квадратная матрица порядка n . Переменной u присвоить значение, равное скалярному произведению строки и столбца (рассматриваемых как векторы), на пересечении которых находится наименьший элемент матрицы (в предположении, что такой элемент единственный).
10. Даны натуральные числа n, m , действительная матрица размера $m \times n$. Найти среднее арифметическое каждого из столбцов, имеющих четные номера.

11. Даны натуральное число n , действительная квадратная матрица порядка n , действительные $a_1, \dots, a_{n-5}$. Элементы последовательности $a_1, \dots, a_{n-5}$ домножить на 3, если наибольший элемент матрицы (в предположении, что такой элемент единственный) расположен на главной диагонали, и на 0.5 в противном случае.
12. В данной действительной матрице размера $m \times n$ поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащий элемент с наименьшим значением. Предполагается, что эти элементы единственны.
13. Дана целочисленная квадратная матрица порядка n . Переменной y присвоить значение наибольшего из элементов матрицы, расположенных на главной диагонали и ниже ее.
16. Дана действительная квадратная матрица порядка n . Рассмотрим те ее элементы, которые расположены в строках, начинающихся с отрицательного элемента. Найти суммы тех из них, которые находятся соответственно ниже, выше и на побочной диагонали.
18. Дана действительная квадратная матрица порядка n . В строках с отрицательным элементом на главной диагонали найти наибольший из всех элементов.
19. Дана действительная матрица размера $m \times n$, все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы этого элемента.
20. Дана действительная матрица размера $m \times n$. Найти среднее арифметическое наибольшего и наименьшего элементов этой матрицы из лежащих на ее побочной диагонали.
21. В данной действительной квадратной матрице порядка n найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.

22. Дана действительная квадратная матрица порядка n . Найти сумму элементов матрицы, расположенных в строках с отрицательным элементом на главной диагонали.
23. Дан двумерный целочисленный массив размером $m \times n$. Некоторый элемент этого массива назовем "седловой точкой", если он является одновременно наименьшим в своей строке и наибольшим в своем столбце. Напечатать номера строки и столбца какой-нибудь "седловой точки" и напечатать число 0, если такой точки нет.
24. Дана действительная квадратная матрица порядка n . Получить целочисленную квадратную матрицу того же порядка, в которой элемент равен единице, если соответствующий ему элемент исходной матрицы больше элемента, расположенного в его строке на главной диагонали, и равен нулю в противном случае.
25. Даны натуральное число m и целочисленная квадратная матрица порядка m . Строку с номером i матрицы назовем отмеченной, если $a_i \leq 0$, и неотмеченной в противном случае. Посчитать число отрицательных элементов, расположенных в отмеченных строках.
26. Дана действительная квадратная матрица порядка n . Построить последовательность действительных чисел $a_1, a_2, \dots, a_n$ по правилу: если в i -той строке матрицы элемент, принадлежащий главной диагонали, отрицателен, то a_i равно сумме элементов i -той строки, предшествующих первому отрицательному элементу; в противном случае a_i равно сумме последних элементов i -той строки, начиная с первого по порядку неотрицательного элемента.
27. Дано натуральное число n . Получить действительную матрицу $\|c_{ij}\|$, для которой
- $$c_{ij} = \frac{i - j}{i^2 + j^2}, \text{ где } i, j = 1, 2, \dots, n.$$
28. Дана действительная квадратная матрица порядка n . В строках с отрицательным элементом на побочной диагонали найти сумму всех элементов.

29. Дан двумерный целочисленный массив размером $m \times n$. Известно, что среди его элементов два и только два равны между собой. Напечатать их индексы.

30. Дано натуральное число n . Выяснить, сколько положительных элементов содержит

$$\text{матрица } a_{ij} \text{ } (i, j = 1, 2, \dots, n), \text{ если } a_{ij} = \begin{cases} \sin \frac{i}{2} \cdot \frac{j}{2} & \text{если } i \text{ и } j \text{ нечетные} \\ 0 & \text{иначе} \end{cases}$$

Лабораторная работа 14

Тема: Объектно-ориентированное программирование в JavaScript

Цель работы: изучить основы объектно-ориентированного программирования в JavaScript

Теоретическая часть

События JavaScript

События, главным образом, инициируются теми или иными действиями пользователя. Если он щелкает по некоторой кнопке, происходит событие Click. Если указатель мыши пересекает какую-либо ссылку гипертекста - происходит событие MouseOver. Существует несколько различных типов событий. Мы можем заставить нашу JavaScript-программу реагировать на некоторые из них. И это может быть выполнено с помощью специальных программ обработки событий. Так, в результате щелчка по кнопке может создаваться выпадающее окно. Это означает, что создание окна должно быть реакцией на событие щелчка - Click. Программа - обработчик событий, которую мы должны использовать в данном случае, называется onClick. И она сообщает компьютеру, что нужно делать, если произойдет данное событие. Приведенный листинг представляет простой пример программы обработки события onClick:

```
<html>
<body>
<form>
<input type="button" value="Click me" onClick="alert('Привет')">
</form>
</body>
</html>
```

В приведенном примере функция alert() позволяет Вам создавать выпадающие окна. При ее вызове Вы должны в скобках задать некую строку.

Опишем синтаксис основных событий, использующихся в JavaScript:

- onClick - Вызывается после щелчка левой кнопкой мыши на объекте.

`<INPUT TYPE="elementType" onClick="function">`

- onMouseOver - Событие происходит, когда указатель мыши помещается над гиперссылкой. `<A HREF="URL" onMouseOver="function">linkText</A>`

- onFocus - Событие происходит в тот момент, когда пользователь переходит к элементу формы select, text или textarea для ввода данных. `<INPUT TYPE="inputType" onFocus="function">`

- onBlur - Событие происходит в тот момент, когда элемент формы select, text или textarea теряет фокус. `<INPUT TYPE="elementType" onBlur="function">`

- onSelect - Обработчик события onSelect вызывается в тот момент, когда выделен текст внутри элемента формы. `<INPUT TYPE="textType" onSelect="function">`

- onSubmit - Событие происходит в момент щелчка мышью на кнопке Submit для отправки данных формы на сервер. `<TAG onSubmit="function">`

- onLoad - Вызывается, когда загрузка документа в окно или в фрейм закончена. `<BODY onLoad="function">`, `<FRAMESET onLoad="function">`

- onUnload - Вызывается, когда пользователь выходит из документа. `<BODY onUnload="function">`, `<FRAMESET onUnload="function">`

- onChange - Событие происходит в тот момент, когда значение элемента формы select, text или textarea изменилось и элемент потерял фокус. `<INPUT TYPE="elementType" onChange="function">`

Приведем некоторые примеры:

Пример 1

```
< html>
< body>
<A HREF="http://www.newmail.ru" onMouseOver="window.status='Бесплатный хостинг';
return true">Ссылка</A>
<form>
<input type="text" size="30" onFocus="window.status='Текст в строке состояния';">
</ form>
< form>
< input type=" text" size="45" value="Впишите свое имя и щелкните по другой строке"
onBlur=" alert('Вы изменили ответ — уверены, что он правильный?');">
</ form>
< form>
< input TYPE=" text" size="45" value="Измените текст и щелкните по другой строке"
onChange=" window. status='Текст был изменен';">
</form>
<form>
<input TYPE="submit" onSubmit="parent.location='thanksalot.html';">
</form></body>
</html>
```

Классы и объекты JavaScript

В языке JavaScript имеется три типа объектов: встроенные объекты, объекты браузера и объекты, которые программист создает самостоятельно

С точки зрения сценария JavaScript браузер представляется иерархически организованным набором объектов. Обращаясь к свойствам и методам этих объектов можно выполнять различные операции над окном браузера, загруженным в это окно документом HTML, а также над отдельными объектами, размещенными в документе HTML. Например, можно создавать новые окна браузера, загружая в них документы HTML, динамически формировать текст документа HTML, обращаться к полям форм, определенных в документе HTML и так далее.

Оборудование и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 7.

Задания

1. Создать форму, которая будет иметь три элемента: поле ввода с просьбой ввести имя, при вводе которого в строке состояния должны появиться слова:

«Впишите сюда свое имя»; два поля для флажков с вопросом о том, что больше нравится пользователю, мороженое или шоколад, которые должны отослать в строку состояния слова: «Вы выбрали...» и выбор пользователя; кнопку отправки данных, при нажатии на которую должно выскочить окно предупреждения, благодарящее пользователя за участие в опросе.

2. Создать документ, в котором располагалась бы кнопка, при нажатии выводящее диалоговое окно, которое открывает второй документ при нажатии на кнопку "Ok" и закрывает данный документ при нажатии на "Cancel"

Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать: название дисциплины, название лабораторной работы, фамилию и инициалы студента, номер группы, фамилию и инициалы преподавателя. Основная часть лабораторной работы должна содержать:

- ☐ вариант вашего задания;
- ☐ распечатку страниц выполненного задания;
- ☐ выводы по проделанной работе.

Защита работы

Защита работы заключается:

- ☐ в выполнении варианта своего задания и оформления отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- ☐ демонстрации преподавателю выполненного задания;
- ☐ в ответах на контрольные вопросы по лабораторной работе;
- ☐ в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Основы работы в Web-среде».

Контрольные вопросы

1. Что такое события в JavaScript?
2. Понятие классы и объекты в JavaScript.
3. Объект Window, его свойства и методы.
4. Объект history, его свойства и методы.
5. Объект location, его свойства и методы.

6. Объект document, его свойства и методы.

Лабораторная работа 15

Тема: Внутренние объекты JavaScript. Потомки объектов JavaScript

Цель работы: изучить принципы работы с внутренними объектами JavaScript

Теоретическая часть

Внутренние объекты не относятся к браузеру или загруженному в настоящее время HTML-документу. Эти объекты могут создаваться и обрабатываться в любой JavaScript-программе. Они включают в себя простые типы, такие как строки, а также более сложные объекты, в частности даты.

Объект array

Примерами объектов-массивов в браузере служат гиперсвязи, метки, формы, фреймы. Массив можно создать одним из следующих способов:

- используя определенную пользователем функцию для присвоения объекту многих значений;
- используя конструктор Array();
- используя конструктор Object().

Объекты-массивы не имеют ни методов, ни свойств.

Пример 1

```
<SCRIPT >
myArray= new Array();
myArray[0]= "first element";
myArray[1]= "second element";
myArray[2]= "third element";
for (var i= 0; i< 3; i++)
{
document.write(myArray[i] + "<br>");
}
</SCRIPT>
```

Объект Date

Объект содержит информацию о дате и времени. Этот объект имеет множество методов, предназначенных для получения такой информации. Кроме того объекты Date можно создавать и изменять, например путем сложения или

вычитания значений дат получать новую дату. Для создания объекта Date применяется синтаксис:

```
dateObj = new Date(parameters)
```

где dateObj - переменная, в которую будет записан новый объект Date.

Аргумент **parameters** может принимать следующие значения:

- пустой параметр, например date() в данном случае дата и время - системные.
- строку, представляющую дату и время в виде: "месяц, день, год, время", например "March 1, 2000, 17:00:00" Время представлено в 24-часовом формате;
- значения года, месяца, дня, часа, минут, секунд. Например, строка "00,4,1,12,30,0" означает 1 апреля 2000 года, 12:30.
- целочисленные значения только для года, месяца и дня, например "00,5,1" означает 1 мая 2000 года, сразу после полночи, так, как значения времени равны нулю.

Данный объект имеет множество методов, свойств объект Date не имеет.

Объект Math

Math - встроенный в JavaScript объект, дающий доступ к константам и математическим функциям. Объект Math делится на две части - свойства, содержащие константы и методы для реализации функций. Свойствами объекта Math являются математические константы:

- E - Константа Эйлера. Приближенное значение 2.718 . . .
- LN2 - Значение натурального логарифма числа два. Приближенное значение 0.693 . . .
- LN10 - Значение натурального логарифма числа десять. Приближенное значение 2.302 . . .
- LOG2_E - Логарифм e по основанию 2
- LOG10_E - Десятичный логарифм e. Приближенное значение 0.434 . . .
- PI - Число ПИ. Приближенное значение 3.1415 . . .
- SQRT2 - Корень из двух

Методы объекта Math представляют собой математические функции:

- abs() Возвращает абсолютное значение аргумента.

- `acos()` Возвращает арккосинус аргумента
- `asin()` Возвращает арксинус аргумента
- `atan()` Возвращает арктангенс аргумента
- `ceil()` Возвращает большее целое число аргумента, округление в большую

сторону. `Math.ceil(3.14)` вернет 4

- `cos()` Возвращает косинус аргумента
- `exp()` Возвращает экспоненту аргумента
- `floor()` Возвращает наибольшее целое число аргумента, отбрасывает десятичную часть
- `log()` Возвращает натуральный логарифм аргумента
- `max()` Возвращает больший из 2-х числовых аргументов. `Math.max(3,5)`

вернет число 5

- `min()` Возвращает меньший из 2-х числовых аргументов.
- `pow()` Возвращает результат возведения в степень первого аргумента

вторым. `Math.pow(5,3)` вернет 125

- `random()` Возвращает псевдослучайное число между нулем и единицей.
- `round()` Округление аргумента до ближайшего целого числа.
- `sin()` Возвращает синус аргумента
- `sqrt()` Возвращает квадратный корень аргумента
- `tan()` Возвращает тангенс аргумента

Объект **String**

Объект **string** представляет собой последовательность символов, ограниченная одинарными или двойными кавычками. Обычно присваивают какой-то переменной строку и используют ее как объект для вызова свойств или методов. Например, `s="internet"`, а свойство `s.length` (длина строки) вернет значение 8. Рассмотрим некоторые методы объекта:

- Большинство методов соответствует тегам HTML: `big()`, `fontcolor(arg)`, `fontsize(arg)`, `small()`, `strike()`, `sub()`, `sup()`

- `anchor` - Выводит строку на экран и делает ее якорем - `textString.anchor(anchorName)`

- blink - форматирует строковый объект в виде мигающей строки - `stringName.blink()`
- bold - Форматирует строковый объект жирным шрифтом - `stringName.bold()`
- charAt - Возвращает символ, находящийся в заданной позиции строки - `stringName.charAt(arg)`
- eval - Вычисляет строку как числовое выражение - `eval(string)`
- fixed - Выводит строку на экран шрифтом фиксированной ширины - `stringName.fixed()`
- italics - Отображает текст курсивом аналогично тегу <i> - `stringName.italic()`
- indexOf - Возвращает позицию символа или подстроки в строке, начиная поиск сначала - `stringName.indexOf()`
- lastIndexOf - Возвращает позицию символа или подстроки в строке, начиная поиск с конца - `stringName.lastIndexOf()`
- link - Создает новую гиперссылку на другой URL - `stringName.link(argument)`
- substring - Позволяет извлечь подстроку длиной `arg2`, начиная с позиции `arg1` - `stringName.substring(arg1, arg2)`
- toLowerCase - Преобразует все символы строки к нижнему регистру - `stringName.toLowerCase()`
- toUpperCase - Преобразует все символы строки к верхнему регистру - `stringName.toUpperCase()`

Оборудование и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 7.

Задания

1. Найти максимальный элемент массива из 5 элементов. Результат вывести по щелчку на кнопке
2. Написать скрипт, содержащий текущее время и дату в текстовых полях время и дата. Попробуйте создать изображение работающих часов

3. Задан текст. Определить позицию первого символа "а" в тексте и вывести на экран строку, начиная с этого символа, длиной 3 символа

Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать: название дисциплины, название лабораторной работы, фамилию и инициалы студента, номер группы, фамилию и инициалы преподавателя. Основная часть лабораторной работы должна содержать:

- ☐ вариант вашего задания;
- ☐ распечатку страниц выполненного задания;
- ☐ выводы по проделанной работе.

Защита работы

Защита работы заключается:

- ☐ в выполнении варианта своего задания и оформления отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- ☐ демонстрации преподавателю выполненного задания;
- ☐ в ответах на контрольные вопросы по лабораторной работе;
- ☐ в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Основы работы в Web-среде».

Контрольные вопросы

1. Что такое внутренние объекты JavaScript.
2. Объект Array.
3. Объект Date.
4. Объект Math.
5. Объект String.

Лабораторная работа 16. Объекты как ассоциативные массивы

Цель работы: научиться разрабатывать скрипты на языке JavaScript, реализующие алгоритмы обработки объектов - ассоциативных массивов.

Теоретический материал по теме

Объекты – это составной тип данных, они объединяют множество значений в единый модуль и позволяют сохранять и извлекать значения по их именам.

Объекты – это неупорядоченные коллекции **свойств**, каждое из которых имеет свои **имя (ключ)** и **значение**. Ключи свойств должны быть строками или символами (обычно строками). Значения могут быть любого типа.

Под **ассоциативным массивом** подразумевают массив, в котором в качестве ключей применяются строки. То есть речь идёт о совокупности пар «ключ-значение». Таким образом, в ассоциативном массиве любое значение связано с конкретным ключом, а доступ к этому значению производится по имени ключа.

Основное отличие простого индексного массива от ассоциативного в том, что в обычном массиве значения хранятся под номерами, которые называются **индексами**, а в ассоциативном - под именами, которые называются **ключами**.

Создание объектов при помощи литералов

Литерал объекта – это выражение, которое создает и инициализирует новый объект всякий раз, когда производится вычисление этого выражения. С помощью единственного **литерала объекта** можно создать множество новых объектов, если этот литерал поместить в тело цикла или функции, которая будет вызываться многократно.

Литерал объекта – это заключенный в фигурные скобки список свойств (пар «имя– значение»), разделенных запятыми.

Ассоциативный массив объявляется так же, как и обычный. Пишется слово `let`, далее прописывается его имя, знак равно, затем, обратите внимание, вместо квадратных пишутся фигурные скобки. В фигурных скобках через запятую располагаются элементы, которые состоят из ключа и значения.

Примеры:

```
let empty = {}; // Объект без свойств
let point = {
  x:0, y:0 };
let circle = { x:point.x, y:point.y+1, radius:2 };
let homer = {
  "name": "Homer Simpson",
  "age": 34,
```

```
"married": true,  
"occupation": "plant operator",  
'email': "homer@example.com"  
};
```

Для удобного восприятия и чтения кода оптимальный вариант формировать массив, где каждый элемент прописан на отдельной строке, а также заключать ключи в кавычки. Имя свойства из нескольких слов должно обязательно заключаться в кавычки.

Доступ к значениям свойств объектов

Для доступа к значениям свойств объекта используется оператор «точка»:

obj.property

Пример:

```
let book = new Object();  
book.title = "Программирование на JavaScript "  
book.chapter1 = new Object(); book.chapter1.title = "Введение  
в JavaScript"; book.chapter1.pages = 11;  
book.chapter2 = { title: "Лексическая структура", pages: 6 };  
alert("Заголовок: " + book.title + "\n\t" + "Глава 1 " +  
book.chapter1.title + "\n\t" + "Глава 2 " +  
book.chapter2.title);
```

Чтобы получить доступ к свойству, можно также использовать и квадратные скобки:

obj["property"].

Квадратные скобки позволяют взять ключ из переменной, например:

obj[varWithKey].

Пример:

```
let user = {  
  name: "John",  
  age: 30,  
  "likes birds": true
```

```

}
// присваивание значения свойству
user["likes birds"] = true;
// получение значения свойства
alert(user["likes birds"]); // true let key = "likes
birds";
// то же самое, что и user["likes birds"] = true; user[key] = true;

```

Перечисление свойств объекта function

```

DisplayPropertyNames(obj) { var names = "";
for(var name in obj) names += name + "\n"; alert(names);
}

```

Проверка существования свойств

```

// Если объект o имеет свойство с именем "x", установить его if ("x" in o)
o.x = 1;
// Если свойство x существует и его значение не равно undefined, установить его.
if (o.x !== undefined) o.x = 1;
o.x = undefined; // свойство x будет существовать, но не будет иметь значения
// Если свойство doSomething существует и не содержит значение null
// или undefined, тогда предположить, что это функция и ее сле-дует вызвать!
if (o.doSomething) o.doSomething();

```

Удаление свойств delete

```

book.chapter2;
// удаление свойства
delete user["likes birds"];

```

При удалении свойства его значение не просто устанавливается в значение `undefined`; оператор `delete` действительно удаляет свойство из объекта.

Цикл `for/in` демонстрирует это отличие: он перечисляет свойства, которым было присвоено значение `undefined`, но не перечисляет удаленные свойства.

Доступ к свойствам объекта возможен также при помощи оператора `[]`, который обычно применяется при работе с массивами:

object.property

object["property"]

При обращении к свойству объекта с помощью нотации массивов [] имя свойства задается в виде строки, которая может создаваться и изменяться во время работы программы.

Пример 1. Код, в котором читаются и объединяются в одну строку свойства address0, address1, address2 и address3 объекта customer:

```
var addr = "";
for(i = 0; i < 4; i++) {
  addr += customer["address" + i] + '\n';
}
```

Пример 2: Программа, обращающаяся к сетевым ресурсам для вычисления текущей суммы инвестиций пользователя на фондовом рынке. Программа разрешает пользователю вводить названия любых имеющихся у него акций, а также количество каждого вида акций. Т.к. пользователь вводит названия акций во время исполнения программы, нет способа узнать имена свойств заранее. Поэтому доступ к свойствам объекта portfolio при помощи оператора «точка» невозможен, необходимо использовать оператор []:

```
var stock_name = get_stock_name_from_user(); var shares =
get_number_of_shares(); portfolio[stock_name] = shares; var
value = 0;
for (stock in portfolio) {
  // Для каждого вида акций в портфеле получаем стоимость
  // одной акции и умножаем ее на число акций.
  value += get_share_value(stock) * portfolio[stock];
}
```

Задания для выполнения

Задание 1. Разработайте скрипты, реализующие алгоритм решения задачи с использованием ассоциативных массивов. Ввод исходных данных осуществлять с использованием формы <form> html-документа. Вывод полученных результатов

осуществлять с использованием функции alert. Сохраните скрипты в файле **script_6.js**.

Задание 2. Создайте по образцу html-документ **lab_6.html** и подключите к нему внешние скрипты **script_6.js**. Для тестирования работоспособности скриптов разработайте по 3 теста (включающих наборы входных и выходных данных). Выполните тестирование и отладку скриптов в браузере.

Варианты индивидуальных задач к Заданию 1:

1. Дан ассоциативный массив, содержащий сведения об учениках школы: имя и фамилия ученика, а также название класса (года обучения и буквы), в котором он учится. Выяснить, имеются ли в школе однофамильцы.
2. Дан ассоциативный массив, содержащий сведения о нескольких автомобилях: марка автомобиля, его номер и фамилия владельца. Найти фамилии владельцев и номера автомобилей данной марки.
3. Дан ассоциативный массив, в котором содержится информация о военнослужащих некоторого войскового подразделения: фамилия, имя, отчество, возраст, рост (от 140 до 210 см). Проверить, имеется ли в данном подразделении хотя бы два человека одного роста.
4. Дан ассоциативный массив, содержащий сведения об экспортируемых товарах: указывается наименование товара, страна, импортирующая товар, и объем поставляемой партии в штуках. Найти страны, в которые экспортируется данный товар, и общий объем его экспорта.
5. Дан ассоциативный массив, который содержит номера телефонов сотрудников учреждения: указывается фамилия сотрудника, его инициалы и номер телефона. Найти телефон сотрудника по его фамилии и инициалам.
6. Дан ассоциативный массив, содержащий информацию о багаже нескольких пассажиров: фамилия пассажира, количество вещей в его багаже и общий вес вещей. Найти пассажира, в багаже которого средний вес одной вещи отличается не более чем на 0.3 кг от общего среднего веса вещи.

7. Дан ассоциативный массив, содержащий сведения о книгах. Сведения о каждой из книг — это фамилия автора, название и год издания. Найти названия книг данного автора, изданных с 1960 года.
8. Дан ассоциативный массив, содержащий сведения о веществах: указывается название вещества, его удельный вес и проводимость (проводник, полупроводник, диэлектрик). Найти удельные веса и названия всех полупроводников.
9. Дан ассоциативный массив, содержащий сведения об игрушках: указывается название игрушки (например, кукла, кубики, мяч, конструктор), ее стоимость в рублях и возрастные границы детей, для которых игрушка предназначена (например, для детей от двух до пяти лет). Вывести на экран названия наиболее дорогих игрушек, цена которых отличается от цены самой дорогой игрушки не более чем на 1 руб.
10. Дан ассоциативный массив, в котором содержится информация о студентах некоторого вуза: указывается фамилия, имя, отчество студента, его пол (мужской, женский), возраст (от 16 до 35), а также номер курса, на котором он учится (от 1 до 5). Узнать номер курса, на котором обучается наибольший процент мужчин.
11. Дан ассоциативный массив, содержащий сведения о кубиках: размер каждого кубика (длина ребра в сантиметрах) и его цвет (красный, желтый, синий или зеленый). Найти количество кубиков каждого из перечисленных цветов и их суммарный объем.
12. Дан ассоциативный массив, содержащий сведения об учениках класса: имя и фамилия ученика, а также отметки, полученные учениками в последней четверти по физике, математике и литературе. Найти учеников, которые по всем предметам имеют отметки не ниже четырех баллов.
13. Дан ассоциативный массив, содержащий сведения о различных датах. Каждая дата — это число, месяц, год. Найти все весенние даты и записать их в файл g.
14. Дан ассоциативный массив, содержащий сведения о клиентах банка: фамилия, имя, отчество клиента, номер его счета и сумма вклада на этом счете. Найти

всех клиентов, имеющих несколько счетов, и вычислить общую сумму вкладов каждого из них.

15. Дан ассоциативный массив, содержащий различные даты. Каждая дата — это число, месяц, год. Найти самую позднюю дату.
16. Дан ассоциативный массив, содержащий сведения о почтовых посылках: фамилия отправителя, фамилия получателя, вес посылки в килограммах и дата ее отправления (число, месяц, год). Найти всех получателей, на имя которых в течение последнего месяца пришло несколько посылок, и вычислить общий вес посылок, полученных каждым из них.
17. Дан ассоциативный массив, содержащий информацию о подписчиках периодических изданий: фамилия подписчика, его домашний адрес, число выписываемых газет и число выписываемых журналов. Найти всех подписчиков, которые выписывают не менее трех газет и двух журналов, и указать их адреса.
18. Дан ассоциативный массив, в котором содержится информация о владельцах дачных участков садоводческого товарищества: фамилия владельца, номер дачного участка и его площадь (в кв. м.). Найти всех владельцев, имеющих несколько дачных участков, указать для каждого владельца номера всех его дачных участков, а также их общую площадь.
19. Дан ассоциативный массив, содержащий сведения о читателях библиотеки: фамилия, имя, отчество человека, год, с которого он является читателем библиотеки, а также номер его читательского билета. Вывести на экран информацию о самом "старом" читателе библиотеке.
20. Дан ассоциативный массив, содержащий сведения о сотрудниках фирмы: фамилия, имя, отчество сотрудника, его возраст и стаж работы в фирме (в годах). Найти самых молодых сотрудников фирмы (имеющих наименьший возраст), стаж работы которых составляет не менее 3 лет.
21. Дан ассоциативный массив, содержащий различные даты: месяц (январь, февраль, март и т. д.) и число (от 1 до 31). Переписать из этого файла все летние даты в файл summer, а все зимнее даты — в файл winter.

22. Дан ассоциативный массив, содержащий сведения об учебниках для средней школы: указываются предмет (например, "Физика"), автор учебника и номер класса, для которого он предназначен (от 1 до 11). Найти предмет, по которому существует наибольшее количество учебников различных авторов для данного класса.
23. Дан ассоциативный массив, в котором содержатся сведения о служащих учреждения: фамилия и инициалы, пол (мужской, женский) и семейное положение. Выяснить, имеется ли вероятность того, что некоторые из сотрудников учреждения состоят в семейном браке.
24. Дан ассоциативный массив, содержащий информацию об абонентах операторов сотовой связи за последний месяц: указываются оператор, номер телефона абонента, число звонков внутри сети и их общая продолжительность. Найти всех абонентов, имевших наибольшее число звонков внутри сети при их наименьшей средней продолжительности.
25. Дан ассоциативный массив, содержащий сведения о вакансиях рабочих мест: указываются требуемая профессия, ежемесячный размер оплаты труда и номер телефона, по которому можно связаться с работодателем. Найти информацию о самых высокооплачиваемых рабочих местах по данной профессии (где предлагаемый ежемесячный размер оплаты труда выше среднего по данной профессии).
26. Дан ассоциативный массив, содержащий информацию о жителях Кемеровской области: фамилия человека, город, в котором он живет и его адрес (название улицы, номер дома и номер квартиры). Найти фамилии двух любых жителей Кемеровской области, живущих в разных городах по одинаковому адресу.
27. Дан ассоциативный массив, в котором содержатся сведения о студентах некоторого вуза: фамилия, имя, отчество, пол, возраст, курс, причем в фамилии, имени и отчестве — не более 12 букв, пол указывается буквами М и Ж, возраст — целое от 16 до 35, курс — целое от 1 до 5. Выяснить, на каком курсе наибольший процент мужчин.

28. Дан ассоциативный массив, содержащий сведения о пассажирах авиакомпании: фамилия, имя, отчество пассажира, номер рейса. Выяснить, имеются ли на данном рейсе однофамильцы, если да, то указать их ФИО.
29. Дан ассоциативный массив, содержащий информацию о имеющихся на складе комплектующих для компьютеров: указывается наименование детали, ее стоимость в рублях и количество таких деталей. От покупателя поступил заказ на приобретение партии комплектующих, в котором указаны их наименования и требуемое число деталей каждого наименования. Выяснить, имеется ли на складе достаточное число комплектующих для выполнения данного заказа. В случае, если заказ может быть выполнен, подсчитать его полную стоимость.
30. Дан ассоциативный массив, в котором содержатся сведения о служащих учреждения: фамилия, имя, отчество, пол (мужской, женский), а также стаж работы в данном учреждении. Найти самые распространенные среди сотрудников учреждения мужское и женское имена.

Лабораторная работа 17.

Работа со строками и регулярными выражениями

Цели работы:

- освоить базовые функции JavaScript по работе со строками;
- ознакомиться с принципами построения регулярных выражений;
- изучить функции JavaScript по работе с регулярными выражениями.

Среда выполнения и отладки:

Текстовый редактор Notepad++, веб-браузер (Firefox, Internet Explorer, Opera или др.).

Теоретические сведения

Определение длины строки

Для получения длины строки можно использовать свойство `length`, которое возвращает длину строки в символах. Для определения числа символов в строке

следует добавить точку после имени переменной, причем за точкой следует свойство `length`: `name.length`.

Изменение регистра строки

JavaScript предлагает два метода перевода целых строк в верхний или нижний регистр.

Метод **`toUpperCase()`** переводит все содержимое строки в верхний регистр. Например, следующий код переводит слово 'Hello' в верхний регистр и выводит его в виде сообщения 'HELLO':

```
var greetings='Hello';  
alert(greetings.toUpperCase());
```

Чтобы перевести всю строку в нижний регистр, следует использовать метод **`toLowerCase()`**:

```
var greetings='Hello';  
alert(greetings.toLowerCase()); // hello
```

Ни один из этих методов на самом деле не изменяет самой строки, сохраненной в переменной, они просто возвращают строку либо только в нижнем, либо только в верхнем регистре. В рассмотренном примере `greetings` по-прежнему содержит «Hello» даже после появления предупреждения.

Поиск в строке: метод **`indexOf()`**

JavaScript предлагает несколько способов поиска слов, чисел или других серий знаков в строках.

Один из методов поиска строки – **`indexOf()`**. Порядок такой: после имени строковой переменной `string` печатается точка, затем **`indexOf()`** и в скобки вставляется искомая строка:

```
string.indexOf('строка, которую надо найти')
```

Метод **`indexOf()`** возвращает число: если искомая строка не найдена, то метод возвращает `-1`.

Когда метод **indexOf()** обнаруживает искомую строку, он возвращает число, равное начальной позиции строки. Рассмотрим пример, который послужит пояснением:

```
var quote = 'быть или не быть';  
var searchPosition = quote.indexOf('Быть'); // возвращает 0
```

В данном случае **indexOf()** ищет фрагмент **быть** в строке **быть или не быть**. Длинная строка начинается с **быть**, поэтому **indexOf()** находит искомый фрагмент в самом начале. Первая позиция считается 0, вторая буква («ы») – 1, а третья (в данном случае «т») – 2.

Метод **indexOf()** начинает поиск с начала строки. Можно искать и с конца строки, используя метод **lastIndexOf()**. Например, в цитате из Шекспира слово **быть** встречается два раза, поэтому можно определить первое **быть** с помощью метода **indexOf()**, а второе – с использованием **lastIndexOf()**:

```
var quote = "быть или не быть";  
var firstPosition = quote.indexOf('быть'); // возвращает 0  
var lastPosition = quote.lastIndexOf('быть'); // возвращает 12
```

Извлечение части строки с помощью метода **slice()**

Чтобы извлечь часть строки, следует использовать метод **slice()**, который возвращает результат извлечения. Например, имеется строка <http://www.sawmac.com> и требуется исключить часть <http://>. Один из способов сделать это — извлечь часть строки, следующей за <http://>, вот так:

```
var url = 'http://www.sawmac.com';  
var domain = url.slice(7); // www.sawmac.com
```

Метод **slice()** требует в качестве аргумента число, которое присваивается индексу первого символа извлекаемой строки. В данном примере (`url.slice(7)`) 7 обозначает восьмую букву в строке (не следует забывать, что первой букве соответствует индекс 0). Данный метод возвращает все символы, начиная со стартовой позиции, исходя из переданного в качестве аргумента индекса и до конца строки.

Также возможно извлечь определенное количество символов из строки, присвоив методу **slice()** второй аргумент. Вот базовая структура метода **slice()** в данном случае:

```
string.slice(start, end);
```

Стартовое значение – это число, указывающее первый знак извлекаемой строки; конечное значение может запутать, так как это не позиция последней буквы извлекаемой строки, а позиция последней буквы плюс 1. Например, если требуется извлечь первые 4 буквы строки **быть или не быть**, следует указать 0 как первый аргумент, а 4 – как второй. Как видно на рис. 3, 0 – это первая буква в строке, а 4 – пятая, но последняя указанная буква не извлекается из строки. Другими словами символ, указанный в качестве второго аргумента, никогда не извлекается из строки.

Замечание. Если требуется извлечь из строки определенное количество знаков можно просто добавить это число к стартовому значению. Например, чтобы вернуть первые 10 букв строки, первый аргумент будет 0 (первая буква) а второй – 0 + 10 или просто 10: `slice(0,10)`.

Также возможно указывать отрицательные числа, например, `quote.slice(-6,-1)`. Отрицательное число считается с конца строки.

Замечание. Если требуется извлечь строку, содержащую буквы, начиная с 6-й с конца и до последней, можно просто опустить конечный аргумент:

```
quote.slice(-6);
```

Регулярные выражения

JavaScript позволяет использовать регулярные выражения, чтобы находить в строках шаблоны. Регулярное выражение – это серия символов, образец шаблона, который требуется найти.

Для создания шаблона используются символы, как *, +, ? и \. Интерпретатор JavaScript сопоставляет их с реальными символами из строки: числами, буквами и т. д.

Для создания регулярного выражения в JavaScript необходимо создать объект, который состоит из серии символов между двумя слэшами. Например, для регулярного выражения, совпадающего со словом `hello`, следует напечатать программный код:

```
var myMatch = /hello/;
```

Существует несколько методов, используемых при работе со строками, которые имеют преимущество над регулярными. Базовый метод – **search()**. Он работает, как метод `indexOf()`, но вместо того, чтобы пытаться найти одну строку в другой, более крупной, он ищет в строке шаблон. Например, требуется отыскать «быть» в строке «быть или не быть». Ранее это делалось с помощью метода `indexOf()`. Можно сделать то же самое с помощью регулярного выражения:

```
var myRegExp = /быть/;           // не надо заключать в кавычки
var quote = 'быть или не быть.';
var foundPosition = quote.search(myRegExp); // возвращает 0
```

Если **search()** находит совпадение, он возвращает позицию первой совпавшей буквы, а если не находит, то возвращает -1. Так, в примере, приведенном выше, переменная `foundPosition` равна 0, поскольку «быть» начинается с самого начала строки (с первой буквы).

Регулярные выражения могут содержать различные символы для обозначения разных типов знаков. Например, точка (.) означает отдельный знак, любой; `\w` соответствует любой букве или числу (но не пробелам или символам \$ или %). В табл. 6 приведены символы, наиболее часто используемые при поиске по шаблонам.

Таблица 6. Символы, используемые в регулярных выражениях

Символ	Значение
.	Один любой символ. Это может быть буква, число, пробел или другой знак.
\w	Любой знак, который может входить в слово. Это может быть буква (в обоих

	регистрах, цифра от 0 до 9, знак подчеркивания (_).
\W	Любой знак, который НЕ может входить в слово. То есть любые знаки, не соответствующие \w.
\d	Любая цифра от 0 до 9.
\D	Любой символ, кроме цифры. Протоположно \d.
\s	Пробел, знак табуляции, знак возврата каретки и знак новой строки.
\S	Все знаки, кроме знака табуляции, знака возврата каретки и знака новой строки. Противоположно \s.
^	Начало строки. Полезно, если обязательным условием является то, что впереди искомой подстроки нет символов (то есть она начинается строку).
\$	Конец строки. Полезно, если обязательным условием является то, что искомая подстрока должна заканчиваться строку. К примеру, /com\$/ соответствует подстрока «com», но только когда она занимает последние три символа строки. То есть /com\$/ соответствует подстроке «com» в строке «infocom», но не в строке «communication».
\b	Пробел, начало строки, конец строки, любой другой символ, не являющийся буквой или цифрой (к примеру, +, = или '). Можно использовать \b для обозначения начала и конца слова, даже если слово находится в начале или конце строки.
[]	Любой символ из помещенных в скобки. К примеру, [aeiou] совпадает с любой из букв в квадратных скобках. Для обозначения последовательностей символов можно использовать тире. Так, запись [a-z] ссылается на любую букву латинского алфавита в нижнем регистре. Запись [0-9] ссылается на цифры, она аналогична \d.
[^]	Все символы, исключая те, которые помещены в скобки. К примеру, [^aeiouAEUIO] ссылается на все символы, кроме гласных букв. [^0-9] ссылается на все символы, не являющиеся цифрами (аналогично \D).
	Сошлется на символ, который справа от , или же на символ слева. К примеру, a b сошлется или на a, или на b, но не на оба сразу.
\	Используется, чтобы «блокировать» служебные символы регулярных выражений (\, *, ., / и т. д.). Это может быть необходимо, если подобный символ должен быть найден в строке. К примеру, «.» в синтаксисе регулярных выражений означает «любой символ» поэтому, если вы хотите включить символ точки в поиск, его нужно «блокировать»: «\.».

Группирование частей шаблона

Можно использовать круглые скобки, чтобы создавать в шаблоне подгруппы. Эти подгруппы очень удобны (если один из символов табл. 7 используется), чтобы найти совпадения со многими различными строками по одному и тому же шаблону.

Таблица 7. Символы для обозначения совпадений

Символ	Совпадает
?	Ноль или одно появление предыдущего символа. Он необязателен, но если он имеется, то должен быть только в одном экземпляре. Например, регулярное выражение colou?r совпадет и с color, и с colour, но не с colourr
+	Одно или больше появлений предыдущего символа. Он должен появиться как минимум один раз
*	Ноль или более появлений предшествующего символа. Он необязателен и может

	являться любое количество раз. Например, <code>.*</code> совпадает с любым количеством знаков
<code>{n}</code>	Точное число появлений предыдущего символа. Например, <code>\d{3}</code> совпадает только с 3 числами в строке
<code>{n,}</code>	n и более появлений предшествующего символа. Например, <code>a{2,}</code> совпадет с буквой a два и более раз: с сочетанием aa в слове <code>aardvark</code> или <code>aaaa</code> в слове <code>aaaahhhh</code>
<code>{n,m}</code>	Предшествующий символ появляется минимум n раз, но не более m раз. То есть <code>\d{3,4}</code> совпадет с 3 или 4 числами в строке (но не с двумя и не с пятью числами в строке)

Совпадение с шаблоном

Метод **search()**, описанный на ранее, – один из способов узнать, содержит ли строка шаблон регулярного выражения. Метод **match()** работает по-другому. Возможно использовать его со строкой, не только чтобы проверить, существует ли в данной строке шаблон – также можно извлечь этот шаблон, чтобы позже использовать его в других скриптах.

Следующий код находит и извлекает URL с использованием **match()**:

```
// Создание переменной, содержащей строку с URL
var text='Мой web-сайт это www.mysite.com';

// Создание регулярного выражения '
var urlRegex = /((\bhttps?:\V\)| (\bwww\.))\S*/

// Нахождение совпадения с регулярным выражением в строке
var url = text.match(urlRegex);

alert (url [0] ) ; // www.mysite.com
```

Сначала код создает переменную, в которой сохраняется строка, включающая URL `www.mysite.com`. Далее задается регулярное выражение для поиска URL. Наконец, к строке применяется метод **match()**. Функция **match()** – это метод для строк, поэтому все начинается с имени переменной, содержащей строку, потом к ней добавляется точка, а затем **match()**. Таким образом методу **match()** передается регулярное выражение для поиска совпадения.

В примере, данном выше, переменная `url` содержит результат совпадения. Если шаблон регулярного выражения не найден в строке, в результате выдается специальное значение JavaScript, называемое **null**. Если совпадение происходит, то скрипт возвращает массив, первым значением которого является совпавший

текст. Например, здесь переменная `url` содержит массив, и в первом элементе массива записан совпавший с шаблоном текст. В данном случае `url[0]` содержит `www.mysite.com`.

Совпадение с каждым образцом шаблона

Метод **`match()`** работает двумя различными способами в зависимости от того, как было настроено регулярное выражение. В примере, приведенном выше, метод возвращает массив с первым образцом совпавшего текста. Так, если имеется длинная строка, содержащая много URL, возвращается только первый найденный URL.

Однако также можно задействовать свойство регулярного выражения **`global`** для поиска в строке более чем одного совпадения.

Поиск становится глобальным, когда добавляется `g` в конец регулярного выражения.

```
var urlRegex = /((\bhttps?:\V)|(\bwww\.))\S*/g
```

Следует обратить внимание, что **`g`** находится за пределами `/` (используемого для закрывания шаблона). Это регулярное выражение осуществляет глобальный поиск. Когда оно используется с методом **`match()`**, то ищет все совпадения в строке и возвращает массив совпавших фрагментов текста.

Возможно переписать код из предыдущего примера, используя глобальный поиск, следующим образом:

```
// Создание переменной, содержащей строку
```

```
// с гиперссылкой
```

```
var text='Не так уж и много замечательных сайтов, подобных  
www.mysite.com и www.mynewsite.com';
```

```
// Создание регулярного выражения
```

```
// с пометкой глобального поиска
```

```
var urlRegex = /((\bhttps?:\V)|(\bwww\.))\S*/g
```

```
// Нахождение совпадения с регулярным выражением в
```

```
// строке
```

```
var url = text.match(urlRegex);
```

```
alert(url[0]); // www.mysite.com
```

```
alert(url[1]); // www. mynewsite.com
```

Можно определить количество совпадений, получив доступ к свойству **length** результирующего массива: `url.length`. В данном примере будет возвращено число 2, поскольку в строке примера было найдено две гиперссылки. Затем возможно получить доступ к каждой совпавшей строке, используя индекс соответствующего ей элемента. Так, в данном примере `url[0]` – это первое совпадение, а `url[1]` – второе.

Задания на лабораторную работу № 3

Задание № 1. Разработка программы, работающей со строковыми данными.

Варианты задания №1

1. Создать программу, формирующую массив строк `A[10]` и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание ‘abc’ (регистр не должен иметь значения);
- вывод на экран первых пяти символов 4-го элемента массива;
- вывод на экран последних трех символов элемента № 7.

2. Создать программу, формирующую массив строк `B[8]` и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание ‘th’ (регистр не должен иметь значения);
- вывод на экран первых двух символов 6-го элемента массива;
- вывод на экран третьего, четвертого и пятого символов элемента № 2.

3. Создать программу, формирующую массив строк `D[10]` и осуществляющую с ним следующие действия:

- вывод массива на экран;

- вывод количества элементов массива, в которых встречается сочетание ‘fff’ (регистр не должен иметь значения);
 - вывод на экран первых трех символов 1-го элемента массива;
 - вывод на экран последних четырех символов элемента № 7.
4. Создать программу, формирующую массив строк R[11] и осуществляющую с ним следующие действия:
- вывод массива на экран;
 - вывод количества элементов массива, в которых встречается сочетание ‘hello’ (регистр не должен иметь значения);
 - вывод на экран первых двух символов 11-го элемента массива;
 - вывод на экран последних трех символов элемента № 3.
5. Создать программу, формирующую массив строк Q[10] и осуществляющую с ним следующие действия:
- вывод массива на экран;
 - вывод количества элементов массива, в которых встречается сочетание ‘text’ (регистр не должен иметь значения);
 - вывод на экран первых четырех символов 6-го элемента массива;
 - вывод на экран последних трех символов элемента № 3.
6. Создать программу, формирующую массив строк S[12] и осуществляющую с ним следующие действия:
- вывод массива на экран;
 - вывод количества элементов массива, в которых встречается сочетание ‘en’ (регистр не должен иметь значения);
 - вывод на экран первых двух символов 7-го элемента массива;
 - вывод на экран символов № 2, 3, 4 второго элемента массива.
7. Создать программу, формирующую массив строк V[9] и осуществляющую с ним следующие действия:
- вывод массива на экран;
 - вывод количества элементов массива, в которых встречается сочетание ‘с++’ (регистр не должен иметь значения);

- вывод на экран первых четырех символов 6-го элемента массива;
- вывод на экран последних трех символов элемента № 5.

8. Создать программу, формирующую массив строк Y[10] и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание 'ru' (регистр не должен иметь значения);
- вывод на экран первых пяти символов 2-го элемента массива;
- вывод на экран последних двух символов элемента № 0.

9. Создать программу, формирующую массив строк Z[9] и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание 'xyz' (регистр не должен иметь значения);
- вывод на экран первых двух символов 2-го элемента массива;
- вывод на экран последних шести символов элемента № 5.

10. Создать программу, формирующую массив строк M[11] и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание 'java' (регистр не должен иметь значения);
- вывод на экран первых шести символов 2-го элемента массива;
- вывод на экран последних трех символов элемента № 0.

11. Создать программу, формирующую массив строк K[10] и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание 'xml' (регистр не должен иметь значения);
- вывод на экран первых пяти символов 4-го элемента массива;
- вывод на экран последних трех символов элемента № 6.

12. Создать программу, формирующую массив строк W[14] и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание 'php' (регистр не должен иметь значения);
- вывод на экран первых шести символов 3-го элемента массива;
- вывод на экран последних трех символов элемента № 8.

13. Создать программу, формирующую массив строк S[10] и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание 'txt' (регистр не должен иметь значения);
- вывод на экран первых пяти символов 7-го элемента массива;
- вывод на экран последних двух символов элемента № 4.

14. Создать программу, формирующую массив строк T[11] и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание 'pdf' (регистр не должен иметь значения);
- вывод на экран первых двух символов 2-го элемента массива;
- вывод на экран последних пяти символов элемента № 4.

15. Создать программу, формирующую массив строк R[10] и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание 'rus' (регистр не должен иметь значения);
- вывод на экран первых трех символов 2-го элемента массива;
- вывод на экран последних двух символов элемента № 7.

16. Создать программу, формирующую массив строк M[12] и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание ‘ехе’ (регистр не должен иметь значения);
- вывод на экран первых четырех символов 7-го элемента массива;
- вывод на экран последних трех символов элемента № 2.

17. Создать программу, формирующую массив строк N[10] и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание ‘jpg’ (регистр не должен иметь значения);
- вывод на экран первых шести символов 2-го элемента массива;
- вывод на экран последних двух символов элемента № 6.

18. Создать программу, формирующую массив строк K[12] и осуществляющую с ним следующие действия:

- вывод массива на экран;
- вывод количества элементов массива, в которых встречается сочетание ‘docx’ (регистр не должен иметь значения);
- вывод на экран первых шести символов 4-го элемента массива;
- вывод на экран последних трех символов элемента № 5.

Задание № 2. Составить программу, которая:

- формирует массив из 10 строк;
- выводит его на экран;
- выводит на экран те элементы, которые являются адресами веб-сайтов.

Задание № 3. Составить программу, которая формирует массив из 10 строк и выводит на экран все элементы, совпадающие с определенным регулярным выражением, вид которого зависит от выбранного варианта

1. Регулярное выражение – почтовый индекс Воронежа – число вида 394xxx – где x – любая цифра.

2. Регулярное выражение – телефон в городе Воронеже (473) 2xx-xx-xx – где x – любая цифра.

3. Регулярное выражение – IP-адрес – значение вида xxx.xxx.xxx.xxx – где x – любая цифра.
4. Регулярное выражение – мобильный номер – значение вида (+79xx) xxx-xx-xx – где x – любая цифра.
5. Регулярное выражение – номер автомобиля – значение вида bxxxbb – где x – любая цифра, b – любая буква.
6. Регулярное выражение – ИНН субъекта из Воронежского региона – значение вида 36xxxxxxxxxx – где x – любая цифра.
7. Регулярное выражение – СНИЛС – значение вида xxx-xxx-xxx xx – где x – любая цифра.
8. Регулярное выражение – ISBN русских изданий – значение вида 978-5-xxxxx-xxx-x – где x – любая цифра.
9. Регулярное выражение – номер паспорта – значение вида xxxx № xxxxxx – где x – любая цифра.
10. Регулярное выражение – время – выражение формата xx:xx:xx – где x – любая цифра.
11. Регулярное выражение – дата – выражение формата dd/mm/yyyy (все значения цифры).
12. Регулярное выражение – почтовый адрес – выражение формата -bbbb@bbb.bb (значения буквы или цифры).
13. Регулярное выражение – шифр группы – выражение формата xx-уу (xx – любые буквы, уу – любые цифры).
14. Регулярное выражение – номер свидетельства о госрегистрации – выражение формата – xx-уу xxxxxx (xx – любые буквы, уу – любые цифры).
15. Регулярное выражение – код типа 32/AX-789 – выражение формата – xx/уу-xxx (xx – любые цифры, уу – любые буквы).
16. Регулярное выражение – код типа A-17(6)/B – выражение формата – х-уу(у)/х (xx – любые буквы, уу – любые цифры).
17. Регулярное выражение – код типа 4593(23)/A+ – выражение формата – xxxx(xx)/у+ или xxxx(xx)/у- (xx – любые цифры, уу – любые буквы).

18. Регулярное выражение – код типа CFR-14/AZ(5) – выражение формата – xxx-yy/xx(y) (xx – любые буквы, yy – любые цифры).

Список литературы

1. Брокшмидт, К. Введение в разработку приложений для Windows 8 с использованием HTML, CSS и JavaScript / К. Брокшмидт. - 2-е изд., испр. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 460 с.
2. Брокшмидт, К. Пользовательский интерфейс приложений для Windows 8, созданных с использованием HTML, CSS и JavaScript / К. Брокшмидт. - 2-е изд., исправ. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 396 с.
3. Информационные Web-технологии / Ю.Ю. Громов / О.Г. Иванова / Н.Г. Шахов / В.Г. Однолько: учебное пособие Электронный ресурс: Тамбовский государственный технический университет, ЭБС АСВ; Тамбов, 2014. - 96 с. - Книга находится в базовой версии ЭБС IPRbooks.
4. Крис, Миллз. Введение в HTML5: учебное пособие / Крис Миллз. - Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 133 с. - Книга находится в базовой версии ЭБС IPRbooks.
5. Савельев, А. О. HTML5. Основы клиентской разработки / А.О. Савельев; А.А. Алексеев. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 272 с.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по организации и проведению самостоятельной работы
по дисциплине «Web-технологии»
для студентов направления подготовки
09.03.01 Информатика и вычислительная техника

Ставрополь 2026

ОБЩАЯ ХАРАКТЕРИСТИКА ДИСЦИПЛИНЫ

Цель и задачи дисциплины

Целями освоения дисциплины являются формирование у студентов универсальных и общепрофессиональных компетенций, основанных на овладении основами web-технологий, интернет-программирования, изучение принципов обмена данными в сети интернет.

Задачи освоения дисциплины:

1. изучение глобальной сети Интернет, принципов ее построения и способов обмена данными;
2. изучение технологий взаимодействия индивидуального и коллективного пользователя с мировыми информационными ресурсами;
3. изучение технологий создания web-сайта.

Место дисциплины в структуре образовательной программы

Дисциплина «Web-технологии» относится к дисциплинам обязательной части.

Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ПК-2. Способен осуществлять проектирование взаимодействия пользователя с системой и проводить эвристическую оценку графического пользовательского интерфейса	ИД-7 _{ПК-2} Использует современные методологии разработки программного обеспечения и технологии программирования	Использует современные методологии разработки программного обеспечения и технологии программирования на языке JavaScript.
	ИД-12 _{ПК-2} Разрабатывает программный код на языках программирования высокого уровня	Разрабатывает программный код с использованием сценарного языка JavaScript.

2. ХАРАКТЕРИСТИКА САМОСТОЯТЕЛЬНОЙ РАБОТЫ

В основу самостоятельной работы студентов по дисциплине положено конспектирование лекций и рекомендуемых источников, подготовка к компьютерному тестированию знаний, оформление и подготовка к защите отчетов по лабораторным работам, выполнению разноуровневых индивидуальных заданий в ходе лабораторных работ. По каждому виду самостоятельной работы предусмотрены определённые формы отчетности,

которые студенты предъявляют преподавателю в ходе лабораторных занятий или в часы индивидуальных консультаций.

Таблица 1 - Технологическая карта самостоятельной работы

Код оцениваемой компетенции, индикатора (ов)	Этап формирования компетенции (№ темы) (в соответствии с рабочей программой дисциплины)	Средства и технологии оценки	Вид контроля, аттестация (текущий/промежуточный)	Тип контроля (устный, письменный или с использованием технических средств)	Наименование оценочного средства
ИД-7 _{ПК-2} ИД-12 _{ПК-2} ИД-1 _{ПК-3}	Тема 1-17	Вопросы для собеседования по лабораторным работам	текущий	устный	Вопросы для собеседования по лабораторным работам
ИД-7 _{ПК-2} ИД-12 _{ПК-2} ИД-1 _{ПК-3}	Тема 1-17	Вопросы для собеседования	текущий	устный	Вопросы для собеседования
ИД-7 _{ПК-2} ИД-12 _{ПК-2} ИД-1 _{ПК-3}	Тема 1-17	Вопросы к экзамену	текущий	устный	Вопросы к экзамену

Для успешного освоения дисциплины, необходимо выполнить указанные в таблице виды самостоятельной работы, используя рекомендуемые источники информации.

1. Перечень основной литературы:

1. Крис, Миллз. Введение в HTML5: учебное пособие / Крис Миллз. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 133 с. - Книга находится в базовой версии ЭБС IPRbooks., экземпляров неограничено

2. Савельев, А. О. HTML5. Основы клиентской разработки / А.О. Савельев ; А.А. Алексеев. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 272 с., экземпляров неограничено

3. Серова, Е. А. Использование web-технологий при создании информационных систем Электронный ресурс / Серова Е. А.: учебно-методическое пособие. - Москва : МИСИ – МГСУ, 2020. - 55 с. - ISBN 978-5-7264-2202-2, экземпляров неограничено

4. Ефромеев, Н. М. Основы web-программирования Электронный ресурс : Учебное пособие / Н. М. Ефромеев, Е. В. Ефромеева. - Саратов : Вузовское образование, 2019. - 128 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4487-0529-8, экземпляров неограничено

5. Брокшмидт, К. Введение в разработку приложений для Windows 8 с использованием HTML, CSS и JavaScript / К. Брокшмидт. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 460 с., экземпляров неограничено

2. Перечень дополнительной литературы:

1. Поляков, Е. А. Web-дизайн Электронный ресурс : Учебное пособие / Е. А. Поляков. - Саратов : Вузовское образование, 2019. - 188 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4487-0489-5, экземпляров неограничено

2. Лучанинов, Д.В. Основы разработки web-сайтов образовательного назначения Электронный ресурс : учебное пособие / Д.В. Лучанинов. - Саратов : Ай Пи Эр Медиа, 2018. - 105 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-4486-0174-3, экземпляров неограничено

3. Основы работы в Web-среде: лабораторный практикум : Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи. Бакалавриат / сост. С. В. Говорова ; ФГАОУ ВО Сев.-Кав. федер. ун-т. - Ставрополь : СКФУ, 2017. - 160 с., экземпляров неограничено

3. Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине (модулю)

1. Информационные технологии : лабораторный практикум : Специальности 10.05.03 - Информационная безопасность автоматизированных систем. 11.03.02 - Инфокоммуникационные технологии и системы связи / сост. С. В. Говорова, М. А. Лапина ; Сев.-Кав. федер. ун-т - Ставрополь : СКФУ, 2017(2016). - 168 с.

2. Основы работы в Web-среде : лабораторный практикум : Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи. Бакалавриат / сост. С. В. Говорова ; ФГАОУ ВО Сев.-Кав. федер. ун-т - Ставрополь : СКФУ, 2017. - 160 с.

3. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ИЗУЧЕНИЮ ТЕОРЕТИЧЕСКОГО МАТЕРИАЛА

При самостоятельном изучении тем лекционных занятий и заданной литературы студентам рекомендуется:

- в день проведения занятий проработать, изученный на занятии учебный материал по конспекту. При этом необходимо выделить вопросы, на которые следует обратить большее внимание при дальнейшем изучении текущей темы. После проработки учебного материала необходимо ответить на, рекомендуемые контрольные вопросы;
- с целью качественного закрепления изученного материала в последующем следует более детально проработать учебный материал с использованием рекомендованной литературы. Студентам рекомендуется

выделить вопросы, требующие более детальной проработки с помощью преподавателя;

- накануне проведения лекции студенту рекомендуется закрепить ранее изученный теоретический материал, подготовить вопросы, требующие уточнения у преподавателя.

Рекомендации по организации работы с литературой

Для овладения, закрепления и систематизации знаний студентам рекомендуется самостоятельная работа с рекомендованной литературой и конспектом лекций. При самостоятельной работе с рекомендованной литературой студентам рекомендуется проводить конспектирование учебного материала, изложенного в рекомендованных источниках.

Конспектирование источников проводится при детальном изучении темы, при этом студентам рекомендуется:

- дополнять конспект лекционного занятия путем его расширения по наиболее важным вопросам, рассмотренным на занятии.

Контроль этого вида самостоятельной работы осуществляется на учебных занятиях путем проверки преподавателем конспекта лекций и собеседования.

При конспектировании источников студенты должны исходить из рационального объема, конспектируемого материала. Конспект должен быть кратким и понятным при его использовании после изучения текущей темы.

Для успешного освоения дисциплины, необходимо самостоятельно детально изучить представленные темы по рекомендуемым источникам информации.

Работа по конспектированию источников оценивается удовлетворительно если:

- конспект имеет достаточный объем детализации по, изучаемой теме, обеспечивающий заданный уровень освоения теоретического материала дисциплины;
- конспект оформлен аккуратно, изучаемый материал изложен последовательно в соответствии с порядком изучения дисциплины, содержит обобщающие выводы по каждой теме.

4. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

При выполнении практикума по дисциплине (лабораторные работы) для достижения базового уровня сформированности компетенций студент должен уметь применять базовые знания по разделам дисциплины, владеть навыками самостоятельного решения учебных задач базового уровня.

При выполнении практикума по дисциплине (лабораторные работы) для достижения повышенного уровня сформированности компетенций студент должен уметь применять повышенные знания по разделам дисциплины, владеть навыками самостоятельного решения учебных задач повышенного уровня.

Методические указания при подготовке к лабораторной работе

Подготовку к лабораторной работе рекомендуется проводить в следующей последовательности:

- уяснить тему и цель, предстоящей лабораторной работы;
- изучить теоретический материал в соответствии с темой лабораторной работы (рекомендуется использовать рекомендованную литературу, конспект лекций, учебное пособие (практикум по лабораторным работам);
- ознакомиться с оборудованием и материалами, используемыми на лабораторной работе (при использовании специализированного оборудования или программного обеспечения необходимо изучить порядок и правила его использования).

Методические указания при выполнении лабораторной работы

При выполнении лабораторной работы студенты должны строго соблюдать, установленные требования безопасности.

При выполнении лабораторной работы студентам рекомендуется:

- уяснить цель, выполняемых заданий и способы их решения;
- задания, указанные в лабораторной работе выполнять в той последовательности, в которой они указаны в лабораторном практикуме;
- при выполнении практического задания и изучении теоретического материала использовать помощь преподавателя;
- оформить отчет по лабораторной работе;
- ответить на контрольные вопросы.

Методические указания при подготовке к защите лабораторной работы

При подготовке к защите лабораторной работы студентам рекомендуется:

- подготовить отчет по лабораторной работе;
- подготовить обоснование, сделанных выводов;
- закрепить знания теоретического материала по теме лабораторной работы (рекомендуется использовать контрольные вопросы);
- знать порядок проведения расчетов (проводимых исследований);
- уметь показать и пояснить порядок исследований при использовании специализированного оборудования или программного обеспечения.

Защита лабораторной работы осуществляется путем собеседования студента с преподавателем. При собеседовании студент представляет на проверку отчет по лабораторной работе. Собеседование со студентом, претендующим на оценку «отлично» проводится по вопросам и практическим заданиям повышенного уровня обучения. Собеседование со студентом, не претендующим на оценку «отлично» проводится по вопросам и практическим заданиям базового уровня обучения.

5. ВОПРОСЫ ДЛЯ СОБЕСЕДОВАНИЯ ПО САМОСТОЯТЕЛЬНОМУ ИЗУЧЕНИЮ ЛИТЕРАТУРЫ

Вопросы (задача, задание) для проверки уровня обученности

1. История возникновения и развития глобальной компьютерной сети Интернет.
2. Основные принципы работы Интернет
3. Основные принципы доступа к Интернет.
4. История создания языка разметки текста.
5. Основные разделы HTML документа
6. Понятие тэга. Основные тэги HTML.
7. Оформление текста в HTML
8. Гипертекстовые ссылки. Структура ссылок в HTML-документе
9. Ссылки на точки внутри документа
10. Графика внутри HTML-документа. Карты в HTML.
11. Списки в HTML
12. Средства описания таблиц в HTML. Создание таблиц в HTML
13. Задание формы — элемент FORM
14. Определение элементов управления формы — тег INPUT. Атрибуты тега INPUT
15. Технология CSS. Подключение таблиц стилей
16. Каскадность стилей. Синтаксис CSS
17. Псевдоклассы
18. Анатомия CSS набора правил
19. Разные типы селекторов (селектором тега, ID селектор, Селектор класса, Селектор атрибута, Селектор псевдокласса)
20. История создания JavaScript, основные понятия и определения. Способы использования JavaScript внутри HTML-документа
21. Использование списков в форме — тег SELECT. Атрибуты тега SELECT
22. Фреймы в HTML.
23. Условные операторы JavaScript
24. Операторы цикла (Оператор for, Оператор for-in, Оператор while)
25. Оператор break .Оператор continue в JavaScript
26. События JavaScript
27. Объект Window в JavaScript
28. Основные методы объекта document в JavaScript
29. Классы и объекты JavaScript
30. Методы объекта History
31. Объект navigator
32. Объект anchor и массив anchors
33. Объект image и массив images
34. Объект link и массив links
35. Массив elements
36. Объект form и массив forms
37. Использование массива forms
38. Основные методы объекта document.
39. Типы сайтов. Принципы web-дизайна

1. Критерии оценивания компетенций (в соответствии с результатами освоения дисциплины)

Оценка **«отлично»** выставляется студенту, если он четко решает задачу, используя язык HTML и технологию CSS для создания web-сайтов. Полно использует язык HTML и технологию CSS для решения задач обработки информации. Грамотно читает коды web-сайтов, понимает основные разделы HTML документа и вносит требуемые изменения в HTML документ.

Оценка **«хорошо»** выставляется студенту, если он достаточно полно решает задачу обработки данных, используя язык HTML и технологию CSS для создания web-сайтов. В основном правильно использует возможности программного обеспечения для создания web-сайтов, использует язык HTML и технологию CSS для решения задач обработки информации. Читает коды web-сайтов, понимает основные разделы HTML документа и вносит требуемые изменения в HTML документ, допуская незначительные ошибки. В основном точно разрабатывает оригинальные HTML документы, пригодные для практического применения. Может допускать незначительные ошибки.

Оценка **«удовлетворительно»** выставляется студенту, если он слабо решает задачу обработки данных, используя язык HTML и технологию CSS для создания web-сайтов. Слабо использует возможности программного обеспечения для создания web-сайтов, использует язык HTML и технологию CSS для решения задач обработки информации. Читает коды web-сайтов, понимает основные разделы HTML документа и вносит требуемые изменения в HTML документ, допуская значительные ошибки. Допускает серьезные ошибки при разработке оригинальных HTML документы, пригодных для практического применения.

Оценка **«неудовлетворительно»** выставляется студенту, если он не решает задачу обработки данных, используя язык HTML и технологию CSS для создания web-сайтов. Не использует возможности программного обеспечения для создания web-сайтов, не использует язык HTML и технологию CSS для решения задач обработки информации. Не может читать коды web-сайтов, не понимает основные разделы HTML документа и не вносит требуемые изменения в HTML документ. С трудом разрабатывает оригинальные HTML документы, пригодные для практического применения.

2. Описание шкалы оценивания

Успеваемость студентов по каждой дисциплине оценивается в ходе текущего контроля и промежуточной аттестации.

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Процедура проведения данного оценочного мероприятия включает в себя:

- отдельное собеседование по каждой из указанных тем;
- составление конспекта.

Предлагаемые задания позволяют проверить компетенции ПК-2, ПК-3

Для подготовки к каждой теме следует изучить и законспектировать рекомендованные источники и научную литературу и на их основании подготовиться

к указанным в методических материалах вопросам для обсуждения.

При проверке задания, оцениваются

- уверенное владение материалом;
- умение четко и логично излагать свои мысли;
- умение творчески подходить к решению основных вопросов темы;
- самостоятельность мышления,
- полное соответствие конспекта установленным требованиям;
- самостоятельная устная речь, полностью раскрывающая суть суть проблематики собеседования.

7.ВОПРОСЫ ДЛЯ СОБЕСЕДОВАНИЯ ПРИ ПРОВЕРКЕ ГОТОВНОСТИ К ЛАБОРАТОРНЫМ РАБОТАМ

Тема: Основы языка HTML.

1. Суть и составные части Web технологии.
2. Основы языка разметки гипертекста - HTML
3. Тэг <HTML>
4. Элемент <BODY>
5. Заголовочные тэги
6. META - метаинформация
7. TITLE - "внешний" заголовок (титул)
8. Комментарии
9. Оформление текста
- 10.Списки

Тема: Гиперссылки. Внедрение изображений

1. Гипертекстовые ссылки
2. Структура ссылок в HTML-документе
3. Ссылки на точки внутри документа
4. Графика внутри HTML-документа
- 5.
6. Тема: Оформление HTML документа. Таблицы
7. Средства описания таблиц в HTML
8. Создание таблиц в HTML

Тема: Формы в HTML

1. Задание формы — элемент FORM
2. Определение элементов управления формы — тег <INPUT>
3. Атрибуты тега <INPUT>
4. Использование списков в форме — тег <SELECT>
5. Атрибуты тега <SELECT>

Тема: Технология CSS. Проектирование оформления web-сайта

1. Технология CSS

2. Подключение таблиц стилей
3. Описание в секции заголовка
4. Вынесение описания стилей во внешний файл
5. Каскадность стилей
6. Синтаксис CSS
7. Псевдоклассы
8. Основные параметры CSS
9. Примеры описания таблицы стилей

Тема: Проектирование структуры web-сайта

1. Типы сайтов
2. Принципы web-дизайна

Тема: Внедрение объектов JavaScript в HTML-документ

1. Способы использования JavaScript внутри HTML-документа
2. Встраивание в тело документа HTML при помощи тега `<SCRIPT>` и `</SCRIPT>`
3. Ссылки на файлы с подгружаемыми скриптами
4. Встраивание с применением функций

Тема: Базовые конструкции языка JavaScript

1. Условные операторы
2. Операторы цикла
3. Оператор for
4. Оператор for-in
5. Оператор while
6. Оператор break
7. Оператор continue

Тема: Основы объектно-ориентированного программирования в JavaScript

1. События JavaScript
2. Классы и объекты JavaScript
3. Объект Window
4. Основные методы объекта document
5. Методы объекта History
6. Объект navigator

1. Критерии оценивания компетенций

оценка «отлично» выставляется, если студент продемонстрировал высокое умение применять полученные знания на практике через решение конкретной задачи по применению web-технологий, свободно без затруднений справился с поставленной задачей, показав владение разносторонними приемами и навыками ее выполнения, не допустил ошибок и неточностей;

оценка «хорошо» выставляется, если студент продемонстрировал умение применять полученные знания на практике через решение конкретной задачи по применению web-технологий, справился с поставленной задачей, показав владение необходимыми приемами и навыками ее выполнения, при этом допустил не более одной ошибки;

оценка «удовлетворительно» выставляется, если студент продемонстрировал посредственное умение применять полученные знания на практике через решение конкретной задачи по применению web-технологий, с трудом справился с поставленной задачей, при этом допустил не более двух ошибок;

оценка «неудовлетворительно» выставляется, если студент не продемонстрировал умение применять полученные знания на практике через решение конкретной задачи по применению web-технологий, не справился с поставленной задачей или допустил при ее решении три и более серьезные ошибки.

2. Описание шкалы оценивания

Успеваемость студентов по каждой дисциплине оценивается в ходе текущего контроля и промежуточной аттестации.

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Процедура проведения данного оценочного мероприятия включает в себя:

- собеседование;
- защита лабораторных работ.

Предлагаемые задания позволяют проверить компетенции ПК-2, ПК-3.

К лабораторному занятию студент должен подготовить ответы на вопросы, проработать дополнительные научные источники, сделать конспект теоретического материала (если это предусмотрено).

Максимальное количество баллов студент получает, если он выполнит лабораторное задание, владеет материалом, умеет логично и четко излагать мысли, показывает самостоятельность выполнения задания.

Основанием для снижения оценки являются:

- слабое знание темы и неполностью выполненное задание;
- невыполненное задание к лабораторной работе;
- отсутствие умения применить теоретические знания для решения лабораторных задач.

При проверке задания, оцениваются

- активное участие в работе;
- уверенное владение материалом;
- умение четко и логично излагать свои мысли;
- умение творчески подходить к решению заданий;
- самостоятельность мышления,
- самостоятельная устная речь, полностью раскрывающая суть сути проблематики собеседования.

8.ВОПРОСЫ К ЭКЗАМЕНУ

Вопросы (задача, задание) для проверки уровня обученности

1. История возникновения и развития глобальной компьютерной сети Интернет.
2. Основные принципы работы Интернет.
3. Основные принципы доступа к Интернет.
4. Адресация.
5. Стек протоколов Интернет.
6. Серверы имен. Понятие зоны, деление DNS-пространства на зоны.
7. История создания языка разметки текста.
8. Основные разделы HTML документа.
9. Понятие тэга. Основные тэги HTML.
10. Гипертекстовые ссылки
11. Структура ссылок в HTML-документе
12. Ссылки на точки внутри документа
13. Графика внутри HTML-документа
14. Средства описания таблиц в HTML
15. Создание таблиц в HTML
16. Задание формы — элемент FORM
17. Определение элементов управления формы — тег <INPUT>
18. Атрибуты тега <INPUT>
19. Использование списков в форме — тег <SELECT>
20. Атрибуты тега <SELECT>
21. Технология CSS
22. Подключение таблиц стилей
23. Описание в секции заголовка
24. Вынесение описания стилей во внешний файл
25. Каскадность стилей
26. Синтаксис CSS
27. Псевдоклассы
28. Основные параметры CSS
29. Примеры описания таблицы стилей
30. Типы сайтов
31. Принципы web-дизайна
32. История создания JavaScript, основные понятия и определения.
33. Способы использования JavaScript внутри HTML-документа (Встраивание в тело документа HTML при помощи тега <SCRIPT> и </SCRIPT>. Ссылки на файлы с подгружаемыми скриптами. Встраивание с применением функций)
34. Условные операторы
35. Операторы цикла (Оператор for, Оператор for-in, Оператор while)
36. Оператор break .Оператор continue
37. События JavaScript
38. Классы и объекты JavaScript
39. Объект Window

- 40. Основные методы объекта document
- 41. Методы объекта History
- 42. Объект navigator
- 43. Объект anchor и массив anchors
- 44. Объект image и массив images
- 45. Объект link и массив links
- 46. Массив elements
- 47. Объект form и массив forms
- 48. Использование массива forms
- 49. Объекты image и массив images
- 50. Загрузка новых изображений.

1. Критерии оценивания компетенций (в соответствии с результатами освоения дисциплины)

Оценка «отлично» выставляется студенту, если студент дал полный, развернутый ответ на поставленные вопросы, при этом показана совокупность осознанных знаний по дисциплине, доказательно раскрыты основные положения вопросов; в ответе прослеживается четкая структура, логическая последовательность, отражающая сущность раскрываемых понятий. Знание демонстрируется на фоне понимания его в системе дисциплины. Ответ изложен литературным языком с использованием технической терминологии. Могут быть допущены недочеты в определении понятий, исправленные студентом самостоятельно в процессе ответа, продемонстрировал высокое умение применять полученные знания на практике через решение конкретной задачи, свободно без затруднений справился с поставленной задачей, показав владение разносторонними приемами и навыками ее выполнения, не допустил ошибок и неточностей.

Оценка «хорошо» выставляется студенту, если студент дал полный, развернутый ответ на поставленные вопросы, при этом показано умение выделить существенные и несущественные признаки, причинно-следственные связи. Ответ четко структурирован, логичен, изложен литературным языком с использованием технической терминологии. Могут быть допущены 2-3 неточности или незначительные ошибки, исправленные студентом с помощью преподавателя, продемонстрировал умение применять полученные знания на практике через решение конкретной задачи, справился с поставленной задачей, показав владение необходимыми приемами и навыками ее выполнения, при этом допустил не более одной ошибки.

Оценка «удовлетворительно» выставляется студенту, если студент дал недостаточно полные и недостаточно развернутые ответы по вопросам билета. Логика и последовательность изложения имеют нарушения. Допущены ошибки в раскрытии понятий, употреблении терминов. Студент не способен самостоятельно выделить существенные и несущественные признаки и причинно-

следственные связи. В ответе отсутствуют выводы. Умение раскрыть значение обобщенных знаний не показано. Речевое оформление требует поправок, коррекции, продемонстрировал посредственное умение применять полученные знания на практике через решение конкретной задачи, с трудом справился с поставленной задачей, при этом допустил не более двух ошибок.

Оценка «неудовлетворительно» выставляется студенту, если ответ представляет собой разрозненные знания с существенными ошибками по вопросу. Присутствуют фрагментарность, нелогичность изложения. Студент не осознает связь обсуждаемых вопросов с другими объектами дисциплины. Отсутствуют выводы, конкретизация и доказательность изложения. Речь неграмотная, гистологическая терминология не используется. Дополнительные и уточняющие вопросы преподавателя не приводят к коррекции ответа студента, ответ на вопросы полностью отсутствует или студент отказался от ответа, не продемонстрировал умение применять полученные знания на практике через решение конкретной задачи, не справился с поставленной задачей или допустил при ее решении три и более серьезные ошибки.

2. Описание шкалы оценивания

Успеваемость студентов по каждой дисциплине оценивается в ходе текущего контроля и промежуточной аттестации.

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Процедура проведения экзамена осуществляется в соответствии с Положением о проведении текущего контроля успеваемости и промежуточной аттестации обучающихся по образовательным программам высшего образования – программам бакалавриата, программам специалитета, программам магистратуры - в СКФУ.

В экзаменационный билет включаются два вопроса.

Для подготовки по билету отводится 20 минут.

При подготовке к ответу студенту предоставляется право пользования рабочей программой.

При проверке практического задания/задачи, оцениваются логичность, точность и аргументированность решения