УЗБЕКСКОЕ АГЕНТСТВО СВЯЗИ И ИНФОРМАТИЗАЦИИ
ТАШКЕНТКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

К защите.
Зав.кафедрой
_____
_____200__г

Выпускная квалификационная работа бакалавра

на тему «Разработка web сайта на основе HTML с использованием JavaScript»

Выпускник_____ _____
(подпись) (Фамилия)
Руководитель_____ _____
(подпись) (Фамилия)
Рецензент_____ _____
(подпись) (Фамилия)
ОТ и ТБ _____ _____
(подпись) (Фамилия)

Ташкент

УЗБЕКСКОЕ АГЕНТСТВО СВЯЗИ И ИНФОРМАТИЗАЦИИ
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Факультет: Информационные технологии
Кафедра: Прикладная информатика
Направления (специальность): 5521900 – Информатика и информационные технологии

У Т В Е Р Ж Д А Ю

Зав кафедрой_____
«_____» _____2008 г.

ЗАДАНИЕ

на выпускную квалификационную работу
Салихов Тимур Абдумаликович
(фамилия, имя, отчество)

1. Тема работы: Разработка web сайта на основе HTML с использованием JavaScript.
2. Утверждена приказом по университету от «11» февраль 2008 г. № 108
3. Срок сдачи законченной работы: 31.05.08 г.
4. Исходные данные к работе: HTML, JavaScript и задачи по курсу "Web программирования.
5. Содержание расчётно – пояснительной записи (перечень подлежащих разработке вопросов): Введения. Web-дизайн и браузеры. Язык разметки гипертекстовых страниц HTML. Обеспечение доступности Web-страницы. Представление текста на Web-страницах. Представление графики на Web-страницах. JavaScript в действии. Управление данными с помощью переменных. Массивы JavaScript и Escape-последовательности. Выражения, условия, операции, строки и числа. Создание сценариев с помощью функций и событий. Описание программ. Руководство пользователя. Заключение. Литература. Приложение.
6. Перечень графического материала: Использование операции вычитания. Главная страница. Проверка правильности электронного адреса. Проверка заполнения полей формы. Объединение всех проверок на уровне формы. Окно примера «Фоновая музыка». Пример оформления цвет текста. Применение динамических стилей, используя JavaScript. Поисковая система. Дополнительные примеры.
7. Дата выдачи задания: 15.01.08.

Руководитель _____ (подпись) Задание принял _____ (подпись)

8. Консультант по отдельным разделам выпускной работы

Раздел	Ф.И.О Руководителя	Подпись дата	
		Задание выдал	Задание получил
Основная часть Охрана труда и техника безопасности	Эргашев А.А.		

9. График выполнения работы

№	Наименование раздела работы	Срок выполнения	Отметка руководителя выполнении
1.	Современные интернет-технологии	15.01.08-26.02.08	
2.	Использование JavaScript	26.02.08-1.04.08	
3.	Разработка учебных программ по предмету "Web программирования"	5.04.08-10.05.08	
4.	Охрана труда и техника безопасности	10.05.08-15.05.08	
5.	Заключение	15.05.08-31.05.08	

Выпускник _____ «____» _____2008 г.
(подпись)

Руководитель_____ «____» _____2008 г.
(подпись)

Мазмуннома

Битирув малакавий ишида «Web дастурлаш» фанининг амалий машғулотларини ўтқазиш учун дастурлар тўплами яратилди, у 5521900 – Информатика ва ахборот технологияси йўналиши бўйича таълим олаётган талабалар учун мўлжалланган.

Аннотация

В выпускной квалификационной работе разработаны примеры программ для проведение практических занятий по предмету "Web программирования" для студентов направления 5521900-Информатика и информационные технологии.

The summary

In final qualifying to work for carrying out of a practical training in a subject "Web programming" examples of programs are developed for students of a direction 5521900-computer science and information technologies.

Содержание

Введение

Термин HTML (Hyper Text Markup Language) означает "язык маркировки гипертекстов". Это понятие более широкое, включает в себя Интернет и локальные сети, редакторы, браузеры, разнообразные програмные продукты, компакт-диски, обучающие курсы, дизайн и многое другое. HTML – своеобразная противоположность сложным языкам программирования, известным только специалистам.

HTML давно перестал быть просто языком программирования. Человек, изучавший этот язык, обретает возможность делать сложные вещи простыми способами и, главное, быстро, что в компьютерном мире не так уж и мало. Гипертекст подходит для включения элементов мультимедиа в традиционные документы. Практически именно благодаря развитию гипертекста, большинство пользователей получило возможность создавать собственные мультимедийные продукты и распространять их на компакт-дисках. Такие информационные системы, выполненные в виде набора HTML-страниц, не требует разработки специальных программных средств, так как все необходимые инструменты для работы с данными (WEB-браузеры) стали частью стандартного программного обеспечения большинства персональных компьютеров. От пользователя требуется выполнить только ту работу, которая относится к тематике разрабатываемого продукта: подготовить тексты, нарисовать рисунки, создать HTML-страницы и продумать связь между ними.

HTML, как основа создания WEB-страниц, имеет прямое отношение и к новому направлению изобразительного искусства – WEB-дизайн. Художнику в Интернете недостаточно просто нарисовать красивые картинки, оригинальный логотип, создать новый фирменный стиль. Он должен еще поместить все это в Сети, продумать связь между WEB-страницами, чтобы все двигалось, откликалась на действие пользователя, поражало воображение, вызывало желание создать что-нибудь свое, оригинальное в этой области.

Целью выпускной квалификационной работы является изучение и анализ вопросов создания Web сайта на основе HTML с использованием JavaScript и

разработать примеры программ для проведение практических занятий по предмету "Web программирования" для студентов направления 5521900-Информатика и информационные технологии.

1. Современные интернет-технологии

1.1. Web-дизайн и браузеры

Многие Web-дизайнеры сходятся во мнении, что одна из главных проблем Web-дизайна – многообразие браузеров и платформ, каждая из которых по-разному поддерживает HTML и сценарии. С выпуском каждого нового браузера улучшаются их характеристики и возможности, но это не означает, что более ранние версии при этом исчезают. В большинстве своем люди не склонны гнаться за новейшим и лучшим. Одни довольствуются тем, что у них имеется, а другие, вероятнее всего, работают на компьютерах фирм или учреждений, которые выбрали браузеры за них.

Как сделать дизайн Web-страницы эстетически и технически интересным, не игнорируя при этом владельцев предыдущих версий браузеров? Неужели Web-страница, рассчитанная на то, чтобы функционировать на любых браузерах, должна быть обязательно скучной? Можно ли угодить всем? И если нет, то где провести черту? Сколько старых версий будет работать с вашей страницей?

В Web-дизайне нет жестких правил. Поскольку главная наша задача – сделать содержимое страницы доступным для максимального количества пользователей, то для продвижения вперед одинаково важны и эксперимент, и использование новых технологий с учетом существующих реалий. Залог успеха дизайнерского решения лежит в понимании потребностей аудитории и в четком представлении, как сайт будет использован.

Браузеры Netscape Navigator и Microsoft Internet Explorer. На рынке доминируют два основных браузера: Netscape Navigator и Microsoft Internet Explorer. Вместе они, включая все их версии, представляет примерно 90 % (или более) используемых сегодня браузеров.

Эти два браузера конкурируют между собой за господство на рынке. Результатом их борьбы стала коллекция фирменных HTML-тегов, а также несовместимые реализации различных технологий (печально известный

Dynamic HTML, а также JavaScript и Cascading Style Sheets – каскадные таблицы стилей). С другой стороны, конкуренция между Netscape и Microsoft в целом способствовала более быстрому развитию среды Web.

Большинство Web-авторов в своей работе ориентируются на Navigator и Internet Explorer, поскольку они занимают львиную долю рынка. Тем не менее, существует ряд других браузеров, которые вы можете принимать во внимание.

В версии Internet Explorer 4.0 для компьютеров Macintosh отсутствует значительная часть функциональных возможностей версии, созданной для Windows, поэтому использование ряда особых возможностей версии 4.0 может исключить из игры некоторых пользователей.

Некоторые документированные различия включают: отсутствие поддержки встраиваемых шрифтов; отсутствие поддержки фильтров CSS и переходов (визуальных эффектов, таких как тени, отбрасываемые объектом, которые используются для элементов текста); отсутствие элементов управления мультимедиа (эффекты переходов и анимации, обычно создаваемые авторскими мультимедийными программными средствами); проблемы с реализацией DHTML.

Несмотря на заявления Microsoft, что DHTML поддерживается всеми платформами, он особенно ненадежен на компьютерах платформы Macintosh.

Браузеры America Online. Пользователи America Online (AOL) используют один из семи возможных браузеров (в зависимости от платформы и версии программного обеспечения AOL), некоторые из них обеспечивают только самую минимальную поддержку HTML.

Последняя версия America Online для PC – это версия 3.0, использует адаптированную версию браузера Microsoft Interne Explorer 3.0. Тем не менее, не всегда можно полностью положиться на эту версию так же, как на стандартный вариант MS Internet Explorer 3.0. (Функциональность особенно ограничена для пользователей компьютеров Macintosh). Многие разработчики Web не раз ужасались, увидев дизайн своего сайта (который великолепно работал в большинстве основных браузеров), после того как он был запущен в

системе AOL и отображен одним из их браузеров.

Проблемы частично возникают из-за того, что AOL полагается на прокси-серверы и методы сжатия изображений. Известно, что используемый AOL метод сжатия изображения имеет проблемы с отображением JPEG-графики, проявляющиеся в появлении пятен и цветных полос. Были отмечены проблемы и при выводе фоновых изображений.

Кроме того, некоторые технологии, такие как Java и Cascading Style Sheets (каскадные таблицы стилей), не доступны для пользователей Windows 3.0 (примерно 40 % пользователей AOL). Владельцы компьютеров Macintosh не смогут использовать JavaScript и ряд других возможностей (примерно 8 % пользователей).

К счастью, создан специальный сайт в помощь тем Web-дизайнерам, которые стремятся сделать свои страницы интересными и доступными для пользователей AOL. Особого внимания заслуживает таблица браузеров, где вы найдете специальный список для каждого из браузеров (по версиям и платформам), перечень технологий и поддерживаемых функций, а также процент сбоев для каждого из браузеров. (Адрес сайта AOL для Web-дизайнеров: http://webmaster.info.aol.com).

WebTV. WebTV приводит в наши квартиры Web через обыкновенный телевизор с пультом дистанционного управления (также, по желанию, можно использовать клавиатуру). Для просмотра Web-страниц WebTV использует собственный специализированный браузер. Он осуществляет синтаксический анализ в соответствии – со стандартом HTML 3.2, но не предоставляет возможностей отображения фреймов, Java, JavaScript, ActiveX или любого другого формата, который требует встраиваемых приложений (за исключением встроенных Shockwave и RealAudio 3.0). Также создано много новых фирменных HTML-тегов, которые используются только в WebTV.

Поскольку WebTV выводит изображение на экран телевизора, предъявляются новые требования к характеристикам цвета и параметрам экрана.

Opera. Opera – это маленький и простенький браузер, созданный норвежской компанией Opera Software в Осло. Этот браузер обладает исключительно малым временем загрузки и минимальными требованиями к объему диска. Достоинством Opera является полное соответствие стандартам HTML. Неточности в написании тегов (например, пропущенные закрывающие теги, неправильное вложение и т. д.), которые пропускают более солидные браузеры, не будут правильно отображаться этим браузером. Opera 5.0 поддерживает Java, каскадные таблицы стилей и DHTML.

Хотя Opera и не стоит на первых местах по частоте использования, но многие разработчики продолжают тестировать свои сайты в Opera, чтобы убедиться в правильности кода.

Lynx. Lynx – это распространяемый бесплатно браузер, обеспечивающий просмотр только текста, предлагает вам быстрый и надежный доступ в Web. Он заслужил известность как наименьший общий знаменатель стандарта, пригодный для тестирования Web-страницы по базовым функциональным характеристикам. Несмотря на простоту, этот браузер не устаревает. Lynx постоянно совершенствуется и модернизируется. Сейчас он обеспечивает поддержку таблиц, форм и даже JavaScript!

Люди действительно используют Lynx, поэтому не стоит удивляться, если клиент закажет разработку сайта для Lynx. Этот браузер также важен для инвалидов по зрению: они используют Lynx вместе с речевыми устройствами.

Будет легче принять решение, какую технологию использовать и где провести черту для обратной совместимости, если знать, какие браузеры используются чаще всего. Наиболее достоверную информацию, конечно, можно получить, ведя статистику посещений сайта.

Отслеживающее программное обеспечение серверов обычно классифицирует посещения по браузерах, осуществляющих запросы. Поэтому, если вы узнали, что только 20 % посетителей вашего сайта используют версии браузеров 4.0, то, может быть, следует повременить с переходом на использование таблиц стилей.

В Интернете можно найти несколько сайтов, предоставляющих статистические данные о браузерах. Статистика на этих сайтах основана на анализе посещаемости самих этих сайтов, что сужает статистическую выборку до узкого круга пользователей, интересующихся такого рода сайтами, – возможно, пользователи, интересующиеся приобретением новых автомобилей или программами телепередач, используют другие браузеры. Статистические данные, помещенные на сайте BrowserWatch, дают подробнейшие сведения о версиях, подверсиях и под-подверсиях каждого отдельного браузера.

1.2. Язык разметки гипертекстовых страниц HTML

Язык разметки гипертекстовых страниц (HTML – Hypertext Markup Language) представляет собой язык, разработанный специально для создания Web-документов. Он определяет синтаксис и размещение специальных инструкций (тегов), которые не выводятся на экран, но указывают браузеру, как отображать содержимое документа. Он также используется для создания ссылок на другие документы, локальные или сетевые, например, находящиеся в сети Интернет.

Стандарт HTML и другие стандарты для Web разработаны под руководством консорциума W3C (World Wide Web Consortium). Стандарты, спецификации и проекты новых предложений можно найти на сайте http://www.3w.org/. В настоящее время действует спецификация HTML 4.0, поддержка которой со стороны основных браузеров постоянно растет.

На практике на стандарт HTML большое влияние оказывает наличие тегов, предложенных и поддерживаемых наиболее известными браузерами, такими как Microsoft Internet Explorer и Netscape Navigator. Эти теги в данный момент могут как входить, так и не входить в состав действующей спецификации HTML.

Информации о тегах HTML Compendium (краткое руководство по HTML) созданно Ron Woodall. Компендиум содержит список тегов и их атрибутов в

алфавитном порядке, а также обновленную информацию о поддержке каждого из них со стороны браузеров. Компендиум HTML находится на сайте http://www.htmlcompendium.org.

Инструментарий редактирования HTML. Документы HTML являются обычными текстовыми ASCII-файлами. Это означает, что для их создания можно использовать любой текстовый редактор, даже с минимальными возможностями. Существуют средства редактирования, разработанные специально для написания HTML. Они позволяют экономить время, так как содержат клавиши быстрого доступа для выполнения повторяющихся операций, например, задания начальных установок документов, таблиц или просто применения стилей к тексту. Редакторы HTML отличаются от авторского WYSIWYG-инструментария (рассматриваемого далее) тем, что требуют знания правил составления HTML вручную, редакторы лишь упрощают и ускоряют этот процесс.

Пользователям Windows определенно следует проверить **HomeSite**, мощный и недорогой редактор HTML компании Allaire Corporation. В нем имеются средства для выделения цветами синтаксических конструкций HTML, функция FTP, контроль синтаксиса и правописания, многофайловый поиск и замещение. Кроме того, он содержит специальные команды и шаблоны для создания более сложных элементов (фреймов, сценариев JavaScript и DHTML). Информация и демонстрационная программа для загрузки находятся по адресу http://www.allaire.com/.

При работе на компьютерах Macintosh обращают внимание на **BBEdit**, коммерческий HTML-редактор компании Bare Bones Software, Inc. Он действительно имеет вес среди Web-разработчиков для для компьютеров Macintosh. В его состав входят удобные и быстрые HTML-инструменты, многофайловый поиск и замена, встроенная FTP-функция, поддержка 13 языков программирования, построитель таблиц, контроль синтаксиса HTML и еще множество функций. Дополнительные сведения и демонстрационную программу можно найти по адресу http://www.bbedit.com.

Авторский инструментарий WYSIWYG. Последние годы характеризуются резким ростом рынка авторских инструментов. HTML-редакторы класса WYSIWYG (What You See Is What You Get – что видишь, то и получишь) имеют графические интерфейсы, которые делают написание HTML больше похожим на программу редактирования текстов или разметки страницы. Первоначальной целью этих программ было освобождение пользователей от тегов HTML, наподобие того, как программы разметки страниц защищают разработчика от набора команд языка PostScript. Сегодня их значимость возросла, так как они повышают эффективность и уровень автоматизации производства документов, обеспечивая в то же время доступ к исходному тексту HTML.

Наиболее популярными в настоящее время WYSIWYG-редакторами являются: Macromedia Dreamweaver, Golive CyberStudio (только для компьютеров Macintosh), Microsoft FrontPage, FileMaker Claris, Home Page, Adobe PageMill.

Теги HTML. Документ HTML содержит текст (содержимое страницы) и встроенные теги – инструкциями о структуре, внешнем виде и функции содержимого. Документ HTML разделяется на две основные части: заголовок – head и тело – body. Заголовок содержит такие сведения о документе, как его название и методическая информация, описывающая содержимое. В теле находится само содержим документа (то, что выводится в окне браузера).

Каждый тег состоит из имени, за которым может следовать список необязательных атрибутов, все они находятся внутри угловых скобок < >. Содержимое скобок никогда не выводится в окне браузера. Имя тега, как правило, представляет собой аббревиатуру его функции, что облегчает его запоминание. Атрибуты являются свойствами, которые расширяют или уточняют функцию тега. Как правило, имя и атрибуты внутри тега не чувствительны к регистру. Тег <BODY BGCOLOR=white> будет работать так же, как <body bgcolor=white>. Однако значения определенных атрибутов могут быть чувствительны к регистру. Это относится, в частности, к именам файлов и

URL.

Контейнеры. Большинство тегов являются контейнерами. Это означает, что у них имеется начальный (открывающий или стартовый) и конечный (закрывающий) теги. Текст, находящийся между тегами, будет выполнять содержащиеся в них инструкции. Например:

The weather is <I>gorgeous</I>today.

Результат: The weather is *gorgeous* today.

Конечный тег имеет то же имя, что и начальный, но перед ним стоит слэш (/). Его можно рассматривать как "выключатель" тега. Конечный тег никогда не содержит атрибутов.

В некоторых случаях конечный тег не обязателен, и браузер определяет конец тега из контекста. Чаще всего опускают конечный тег <p> (абзац). Браузеры раньше поддерживали этот тег без соответствующего завершения, поэтому многие авторы Web привыкли использовать краткую форму. Это разрешено не всем тегам, и не все браузеры прощают их отсутствие. Поэтому, если есть сомнения, включите в текст закрывающий тег. Это особенно важно, когда в документе вы используете каскадные таблицы стилей.

Автономные теги. Некоторые теги не имеет завершающих тегов, потому что они используются для размещения отдельных (автономных) элементов на странице. Одним из них является тег изображения <img>, он просто помещает графику в поток страницы. Другие автономные теги – это разрыв строки (
), горизонтальная линия (<hr>) и теги, содержащие информацию о документе и не влияющие на содержимое, выводимое на экран, такие как <meta> и <base>.

Атрибуты. Атрибуты добавляются в тег для расширения или модификации его действий. К одному тегу можно добавить несколько атрибутов. Если атрибуты тега следуют после имени тега, они разделяются одним или несколькими пробелами. Порядок следования не важен. Большинство атрибутов имеют значения, которые следуют за знаком равенства (=), находящимся после имени атрибута. Длина значений ограничена 1024 символами. Значения могут быть чувствительны к регистру. Иногда значения

должны находиться в кавычках (двойных или одинарных). Правила записи значения следующие:

– если значение представляет собой одно слово или число и состоит только из букв (a-z), цифр (0-9) и специальных символов (точка <.> или дефис <->), то можно поместить его после знака равенства без кавычек;

– если значение содержит несколько слов, разделенных запятыми или пробелами, или содержит специальные символы, отличные от точки или дефиса, тогда его необходимо поместить в кавычки. Например, URL требуют кавычек, потому что они содержат символы "://". Также кавычки необходимы при задании значений цветов с использованием формата "#rrggbb".

Если вы не уверены, стоит ли использовать кавычки, используйте их всегда для всех значений.

В теги HTML могут помещаться другие HTML-теги для осуществления воздействия нескольких тегов на один элемент. Это называется вложением, и, что бы правильно его осуществить, начальный и конечный теги вложенного тега должны обязательно находиться между начальным наконечным тегами внешнего тега, например:

The Weather is <B><I>gorgeous</I></B> today.

Результат: The weather is ***gorgeous*** today.

Часто встречающейся ошибкой является перекрытие тегов. Хотя часть браузеров отображают содержимое, отмеченное таким образом, многие не разрешают нарушать правило, поэтому важно размешать теги правильно. Следующий пример показывает неверное вложение тегов (заметьте, что тег <B> закрывается перед закрытием <I>):

The weather is <B><I>gorgeous</B></I>today – данная информация, игнорируемая браузерами.

Информация, игнорируемая браузерами. Некоторая информацияНиже приводится информация, содержащаяся в документе HTML, включая определенные теги, которая будет игнорироваться при просмотре браузерами. В ее состав входят:

– разрывы строк. Символы конца строк в документе HTML игнорируются. Текст и элементы будут переноситься до тех пор, пока в потоке текста документа не встретится тег <p> или
. Разрывы строк выводятся, если текст обозначен как текст с заданным форматом (<pre>);

– символы табуляции и множественные пробелы. Когда браузер встречает в документе HTML символ табуляции и несколько последовательных символов пробела, он выводит только один пробел. Таким образом, если документ содержит: "far, far away", браузер выведет "far, far away". Дополнительные пробелы можно добавить в текстовый поток, используя символ неразрывного пробела (Snbsp;). Кроме того, все пробелы выводятся, если текст является форматированным (находится в тегах <pre>);

– множественные <p>-теги. Последовательность тегов <p>, не прерываемых текстом, всеми браузерами интерпретируется как избыточная. Содержимое будет выводиться так, как если бы был только один тег <p>. Большинство браузеров выведет несколько тегов
 в виде нескольких переходов на новую строку;

– нераспознаваемые теги. Если браузер не понимает тег или тот был неверно задан, то браузер его просто игнорирует. В зависимости от тега и браузера это может привести к различным результатам. Либо браузер ничего не выведет, или он может отобразить содержимое тега как обычный текст;

– текст в комментариях. Браузеры не выводят текст между специальными элементами <! и ->, которые используются для обозначения комментариев. После символов начала комментария и перед символами окончания обязательно должен находиться пробел. В сам комментарий можно помещать практически все. Комментарии нельзя вкладывать. В Microsoft Internet Explorer имеется фирменный тег, обозначающий комментарии <comment>...</comment>. Однако, он не поддерживается другими браузерами.

1.3. Обеспечение доступности Web-страницы

При разработке Web-страницы фиксированного размера, вероятно, придется выбирать для нее размер экрана. Здравый смысл подсказывает, что страница должна быть доступна (и правильно отображаться) для максимально возможного числа пользователей. Идея проста: необходимо определить наиболее часто используемое разрешение дисплея и разработать страницу таким образом, чтобы страница гарантированно заполняла все рабочее пространство.

Большинство дизайнеров рекомендуют разрабатывать страницы в формате 640x480, чтобы при просмотре пользователям не пришлось применять горизонтальную прокрутку. Горизонтальная прокрутка всегда затрудняет восприятие, поэтому дизайнеры традиционно ее отвергают.

Все большее число разработчиков считает стандартным разрешение 800x600. И совсем единицы разрабатывают страницы для еще более высоких разрешений. Конечно, ваше решение будет, в первую очередь, зависеть от аудитории. Например, если сайт ресурсов для дизайнеров графики, то считаем, что они имеют дисплеи, по крайней мере, с разрешением 800x600 или выше, в соответствии с чем и разрабатывается страница. Если сайт предназначен специально для WebTV или какого-то другого устройства отображения, следует ориентироваться на это конкретное устройство.

Достойный уважения Web-дизайн включает разработку страниц, доступных для пользователей с ограниченными возможностями, в частности по зрению и слуху. Консорциум World Wide Web объявил об инициативе Web Accessibility Initiative (WAI), которая ставит целью сделать Web более доступным для всех пользователей. Однако успех данной инициативы зависит от участия в ней рядовых разработчиков, которые могут (или не могут) создать Web-сайты в соответствии с поставленными задачами.

Пользователи с ограниченными возможностями зрения могут использовать специальные устройства для увеличения изображения,

находящегося на экране. В этом случае к дизайну не предъявляется никаких специальных требований. Многие люди с проблемами зрения используют текстовые браузеры (такие как Lynx) вместе с программным обеспечением, которое громко читает содержимое страницы. В любом случае основное внимание уделяется структуре документа и его тексту. Графическое содержимое может быть просто утеряно.

Средства HTML 4.0. Спецификация HTML 4.0 вводит ряд новых атрибутов и тегов, созданных специально для того, чтобы сделать Web-документы доступными для более широкого круга пользователей. Кратко перечислим некоторые новые возможности HTML 4.0. (Расширенный список возможностей размещен на сайте http://www.w3.org/WAI/References/HTML4-access, а полные спецификации данной версии – на сайте http://www.w3.org/TR/REC-html40).

HTML 4.0 предлагает следующие новые возможности, обеспечивающие доступность:

– дальнейшее разделение структуры документа и его внешнего представления. Информацию о стиле HTML 4.0 предлагает размещать в каскадных таблицах стилей;

– навигационная помощь, например, клавиши доступа и индексирование порядка табуляции для доступа к элементам страницы с использованием только клавиатуры;

– рекомендации, касающиеся новой клиентской карты-изображения, объединяющей графические и текстовые ссылки;

– новые теги <abbr> и <acronym>, которые помогают речевым и другим устройствам интерпретировать аббревиатуры и акронимы;

– возможность логически группировать строки и столбцы таблиц, снабжать их заголовками, резюме и длинными описаниями содержимого, облегчая интерпретацию таблиц;

– возможность группировать элементы управления формами и создавать длинные списки выбора, более ясные для восприятия. Элементы форм также

доступны через клавиши табуляции и быстрого доступа;

– улучшенный механизм создания альтернативного текста. Атрибут alt теперь обязателен для тега <img>. Чтобы обеспечить связь с более длинными текстовыми пояснениями к изображениям, введен атрибут longdesc.

Для добавления информации о любом элементе, можно использовать атрибут title.

Средства CSS. Каскадные таблицы стилей или CSS (от английского Cascading Style Sheets) являются следствием дальнейшего развития HTML и дают нам возможность перейти на следующий уровень представления информации. Таблицы стилей позволяют разделить смысловое содержимое странички и его оформление.

В первых версиях стандарта HTML не было предусмотрено никаких средств для управления внешним видом информации. Общая концепция гипертекста была направлена на доступность информации для любых устройств, способных воспроизводить текст. Для разметки рекомендовалось использовать только логические теги, определяющие заголовки, подзаголовки, списки, абзацы, цитаты и т.д. – то есть, те элементы, которые и составляют структуру документа. Интерпретация же внешнего вида оставалась полностью на совести оконечного терминала.

Однако с тех пор много что изменилось, и стандарт HTML потерял первоначальную стройность. Вначале Netscape добавил "улучшенные теги", которые позволили более широко управлять внешним видом представляемой информации. Нововведение прижилось, и все расширения Netscape стали стандартом de facto. Потом точно также поступила Microsoft. Когда спохватились, то HTML представлял собой ужасную смесь логических и оформительских тегов, несовместимых расширений и полностью перестал отвечать первоначальной концепции – представлять информацию на любом устройстве независимо от его характеристик по выводу информации.

Тогда была предпринята широкомасштабная стандартизация. В результате чего на свет явился стандарт HTML 3.2. Он не был революционным,

а лишь расставил по местам все нововведения и выработал общие рекомендации для производителей браузеров. Революционные изменения были введены в новом стандарте – HTML 4.0 или, как его стали называть, Dynamic HTML. В обращение были введены слои, таблицы стилей и универсальная объектная модель браузера.

В новом стандарте попытались вернуться к истокам концепции HTML. Четвертая версия, как и первая, рекомендует создавать странички таким образом, чтобы они могли быть воспроизведены на любом устройстве – будь это 21" дисплей или маленький черно-белый экран сотового телефона.

Каким же образом была решена проблема с представлением внешнего вида информации? В этом и заключается революционность подхода. Все оформление рекомендуется вынести во внешний стилевой файл. Основная же страничка будет содержать только информацию и ссылки на необходимые стили.

При показе странички конкретному устройству должна быть задействована соответствующая случаю таблица стилей. Для сотового телефона и дисплея компьютера они, разумеется, должны быть разными. В первом случае мы используем минимальное оформление, которое позволит представить информацию наиболее оптимально и компактно. Во втором же случае в нашем распоряжении имеется все богатство шрифтового и цветового оформления.

Таблицу стилей нужно написать всего один раз при создании сайта для каждого из устройств, на котором планируется вывод информации. К тому же таблица стилей может быть единой для целого сайта. И, следовательно, не нужно будет повторять одни и те же описания стилей на каждой из страниц.

Размещение всей стилевой информации в одном внешнем файле открывает нам и другие полезные возможности – ведь изменив содержимое только одного (!) стилевого файла, мы можем в считанные секунды сменить весь дизайн сайта. Причем никаких других переделок не понадобится. Разумеется, это верно лишь в том случае, если первоначально сайт был

спроектирован верно.

CSS2 (Cascading Style Sheets, Level 2) – самая последняя рекомендация по каскадным таблицам стилей, предоставляет механизмы для улучшенной интерпретации страниц неграфическими и не визуальными устройствами. Усовершенствования включают:

– механизмы, с помощью которых созданная пользователем таблица стилей может заменить все таблицы стилей более высоких уровней в каскаде. Это дает конечному пользователю возможность полностью управлять отображением. Пользователь получает возможность создавать настраиваемые таблицы стилей для вывода страниц в соответствии со специальными требованиями;

– специализированная поддержка для загружаемых шрифтов – таким образом уменьшается тенденция помещать текст в графику для улучшения внешнего вида страницы;

– механизмы позиционирования и выравнивания, которые отделяют содержимое от внешнего представления. Эти таблицы стилей должны исключить некорректное использование тегов HTML для создания особых эффектов отображения. Теги HTML можно использовать для логической структуризации документа, делая его более простым для интерпретации не визуальными посредниками;

– средства управления для звукового вывода доставленной по Web информации;

– улучшенные средства навигации, такие как цифровые маркеры, которые можно добавлять в документ в целях ориентации.

Источники разработки доступных страниц

1). Официальный сайт WAI. На нем есть ряд полезных ссылок к источникам, связанным с проблемой доступности (http://www.w3.org/WAI).

2).Очень хороший источник статей и руководств для авторов HTML расположен на сайте Фонда слепых Юрия Рубинского, который является своеобразным оружием борьбы за права инвалидов на расширенный доступ к

технологиям (http://www.yun.org/Webable).

1.4. Представление текста на Web-страницах

При создании профессиональной графики для Web используется текст со сглаженными краями. Сглаживание – это легкая размытость на неровных краях, сглаживающая переходы между цветами. Не сглаженные края, напротив, выглядят зазубренными и ступенчатыми. Исключением из этого общего правила является текст очень малого размера, (10 пунктов и меньше), применение сглаживания делает его практически неразличимым. Текст малых размеров будет выглядеть намного лучше без сглаживания.

Два комплекта шрифтов. При разработке Web-страницы средствами базового HTML есть два комплекта шрифтов; пропорциональный и шрифт фиксированной ширины. Проблема заключается лишь в том, что неизвестно, какой из них и какого размера будет использован при отображении.

Пропорциональный шрифт – иначе "шрифт переменной ширины" для каждого символа выделяет разное количество места в зависимости от его начертания. Например, в пропорциональном шрифте заглавная "W" занимает больше места в строке по горизонтали, чем прописная "I". Такие гаринитуры, как: Times, Helvetica и Arial являются примерами пропорциональных шрифтов.

Web-браузеры для большинства текстов на Web-странице, включая основной текст, заголовки, списки, цитаты и т. д., используют пропорциональные шрифты. Как правило, большие отрывки основного текста удобнее читать, когда они напечатаны пропорциональными шрифтами. Поскольку большинство пользователей не имеют времени заменить шрифты, установленные по умолчанию, с большой вероятностью можно предположить, что текст на вашей странице будет отображен шрифтом Times размером 10 или 12 пунктов (по умолчанию в Netscape) или Helvetica (по умолчанию в Microsoft Internet Explorer). Но это всего лишь общее правило.

Шрифт с фиксированной шириной предоставляет одинаковое место для

всех символов шрифта. Заглавная "W" занимает не больше места, чем прописная "I". Примерами шрифтов фиксированной ширины являются гарнитуры Courier и Monaco. В Web-браузерах шрифты фиксированной ширины используются для отображения любого текста внутри следующих HTML-тегов: <pre>, <tt>, <code>, <kbd>,<samp>, <xtp>.

Поскольку многие люди не меняют настройку шрифтов, установленную по умолчанию, текст, находящийся в указанных тегах, будет выведен одним из шрифтов типа Courier.

Текст в изображениях. Дизайнеры быстро поняли, что самый верный способ абсолютного контроля над шрифтами – поместить текст в изображение. Можно часто видеть заголовки, подзаголовки и объявления, выполненные в виде файлов GIF. Многие Web-страницы представлены исключительно в графике, которая содержит внутри себя весь текст страницы.

Преимущества использования графики вместо HTML-текста абсолютно очевидны:

– можно определять тип шрифта, размер, интерлиньяж, промежуток между буквами, цвет и выравнивание – все атрибуты, которые вызывают сложности только в HTML;

– ваша страница будет одинакова при выводе во всех графических браузерах.

Но у этого метода имеется ряд недостатков:

– изображение загружается дольше, чем текст, т.к. графические файлы обычно на несколько порядков больше, чем HTML-тексты, имеющие то же содержание;

– в неграфических браузерах содержание утрачивается. Пользователи, которые не могут (или не хотят) просматривать графику, не увидят и текста. Альтернативный текст (используется атрибут Alt) на месте графического изображения помогает, но его возможности ограничены и это не всегда надежный способ отождествления отсутствующей графической информации;

– информацию, находящуюся в изображении, нельзя индексировать или

организовать ее поиск. В результате исключаются из документа важные части информации.

Размер шрифта. Обычно размер шрифта определяется в пунктах (72 пункта (пт) = 1 дюйм высоты шрифта) но, к сожалению, эти размеры не достаточно точно переводятся между платформами. Отчасти это происходит потому, что их операционные системы управляют дисплеями с различными разрешениями. Обычно Windows использует разрешение экрана 96 точек/дюйм, а MacOS – 72 точек/дюйм. Мониторы MultiScan допускай более высокое разрешение.

Шрифт на экране дисплея Macintosh имеет точно такой же размер, как и при печати (например, 12 пт Times на экране выглядит так же, как 12 пт Times на бумаге).

Для шрифтов Microsoft подобное соглашение не выполняется, и размер шрифта при выводе на экран больше, что облегчает чтение с дисплея. В результате шрифт размером 12 пт на Windows больше похож на печатный шрифт в 16 пунктов. Чтобы получить на Windows печатный размер 12 пт, вам нужно выбрать размер шрифта 9 пунктов (но тогда пользователи компьютеров Macintosh увидят текст почти неразборчивым, так как он будет отображен шрифтом размером всего 6,75 пт).

1.5. Представление графики на Web-страницах

На данный момент почти все изображения в Web, представлены в двух форматах: GIF и JPEG. Третий соперник, заслуживающий упоминания, формат PNG, борется за поддержку и внимание браузеров. Далее – краткий обзор "большой тройки" онлайновых графических форматов. Более подробная информация представлена в главах, посвященных каждому из форматов.

GIF. GIF – Grafic Interchange Format можно назвать традиционным форматом файлов Web. Он был первым форматом файлов, который поддерживался Web-браузерами, и по сей день продолжает оставаться

основным графическим форматом Web.

Его свойства состоят в следующем:

– поддерживает не более 256 цветов (меньше можно и часто нужно);

– использует палитру цветов;

– использует сжатие без потери информации по методу LZW (который подобен применяемому в архиваторе PKZIP, и, следовательно, GIF-файлы в дальнейшем практически не сжимаются);

– поддерживает чересстрочную развертку;

– является поточным форматом, т.е. показ картинки начинается во время перекачки;

– позволяет назначить одному из цветов в палитре атрибут прозрачный, что применяется при создании так называемых прозрачных GIFов;

– имеет возможность сохранения в одном файле нескольких изображений, что находит свое применение при изготовлении анимированных GIFов;

– поддерживает возможность вставки в файл управляющих блоков, которые позволяют вставлять комментарии в файл (например, об авторских правах), осуществлять задержку между показами изображений и т.д.

А теперь немножко разъяснений – к чему эти свойства могут привести. Как мы написали, GIF поддерживает не больше 256 цветов, а это значит, что все изображения, которые мы сохраняем в GIF-формате, явно или неявно уменьшают количество цветов, чтобы уложиться в этот лимит (разные программы с разным успехом). А отсюда вывод – если у взять красивую фотографию с плавными переходами и едва уловимыми оттенками цвета, то после преобразования все будет гораздо хуже – оттенки перестанут быть неуловимыми, и вся фотография приобретет неестественный, нереалистичный вид. Поэтому, если надо все-таки сохранить фотографию в формате GIF и передать все оттенки, то приходится идти на хитрости. Например, к фотографии можно применить какой-нибудь художественный фильтр и превратить ее в рисунок или применить тонирование. Зато нет никаких

проблем с сохранением рисунков и чертежей в этом формате, они, как правило, хорошо сжимаются и не содержат много цветов.

JPEG. Вторым наиболее популярным графическим форматом в Web является JPEG – Joint Photographic Experts Group. Он содержит 24-разрядную информацию о цвете. Это 16,77 млн цветов в отличие от 256 цветов формата GIF. В JPEG используется так называемое сжатие с потерями. Это означает, что некоторая информация об изображениях в процессе сжатия отбрасывается, но в большинстве случаев ухудшение качества изображения не наносит вреда и часто даже не заметно.

Фотографии или любые изображения с плавными градациями цветов лучше всего сохранять в JPEG-формате, потому что он предлагает более высокое качество изображений, умещающихся в файл меньшего объема. Тем не менее, JPEG не является лучшим решением для графических изображений с одноцветными областями, поскольку этот формат имеет тенденцию испещрять цвета крапинками и конечный файл, как правило, будет несколько больше, чем GIF-файл для того же изображения.

PNG. Есть еще третий графический формат, конкурирующий за постоянное использование в Web. Это формат PNG – Portable Network Graphic, который, несмотря на некоторые достоинства, с 1994 г. находится более или менее в тени. Только недавно браузеры начали поддерживать PNG как встроенную графику, но PNG имеет все шансы стать очень популярным форматом в Web. Именно поэтому он включен здесь в "большую тройку". PNG может поддерживать 8-разрядные индексированные цвета, 16-разрядные полутона или 24-разрядные полноцветные изображения, используя схему сжатия без потерь. Это обеспечивает более высокое качество изображений, а иногда и меньший объем файлов по сравнению с форматом GIF. Кроме того, файлы PNG имеют некоторые замечательные функции, например, встроенное управление коэффициентом гамма, и изменяемые уровни прозрачности (это позволяет показывать рисунок фона сквозь отбрасываемые мягкие тени).

Разрешение и размер файла изображений. Поскольку изображения Web существуют только на экране дисплея, будет технически правильно измерять их разрешение в пикселах на дюйм (ppi – pixels per inch). Другая единица измерения разрешения – количество точек на дюйм (dpi – dots per inch) относится к разрешению печатных изображений и зависит от разрешения печатающего устройства.

Но, так как реальные размеры графики зависят от разрешения дисплея, измерение в дюймах становится для Web-окружения неприемлемым. Единственной значимой единицей измерения становится пиксел.

Практично создавать изображения с разрешением 72 ppi (это лучший вариант для представления на экране), обращая внимание только на общие размеры в пикселах. В процессе создания графики на Web можно вообще не использовать дюймы. Важен размер изображения по сравнению с другими изображениями на странице и общего размера окна браузера.

Например, многие пользователи по-прежнему используют 14-дюймо-вые дисплеи с разрешением 640x480 пикселов. Чтобы гарантировать заполнение графической заставкой всего пространства экрана, лучше сделать его шириной не более 600 пикселов (учитывая, что часть пикселов справа и слева будет использована для окна и для полосы прокрутки). Размер остальных кнопок и изображений на странице следует измерять в пикселах относительно банера, имеющего ширину 600 пикселов.

Размер файла. Без сомнения, именно графика сделала Web таким, каким мы его видим сегодня, но как дизайнер вы должны знать, что многие пользователи испытывают к графике в Web чувство на грани любви и ненависти. Не стоит забывать, что графика увеличивает время, необходимое Web-странице для передачи по сети; большой объем графики означает существенное время загрузки, которое испытывает терпение читателя, особенно если он дозванивается с использованием стандартного модемного соединения.

В этом отношении для Web-дизайнера существует единственное наиболее

важное правило: размер файла графического изображения должен быть минимально возможным! Создание изображений, предназначенных для передачи по сети, возлагает ответственность на разработчиков серьезно относиться к проблеме времени загрузки.

2. Использование Java Script

2.1. JavaScript в действии

JavaScript - язык подготовки сценариев, позволяющий сделать Web-страницы более интерактивными и функциональными. После его изучения можно заниматься разработкой Web-приложений на качественно новом уровне. Для написания сценариев JavaScript обязательно требуется знание языка HTML.

Здесь приводится следующая информация:

- важность изучения JavaScript;
- история JavaScript;
- некоторые различия между языками подготовки сценариев и языками программирования; а инструменты, необходимые для написания сценариев JavaScript.

Присмотритесь к сайтам, которые вы посещаете. На каждой странице содержится текст и некоторое количество картинок. Быть может, есть часы, показывающие время, или бегущий текст в строке состояния браузера. Возможно, на сайте имеется форма, которую нужно заполнить. Если пропустить какую-либо графу анкеты, то появится сообщение об ошибке. На некоторых страницах встречаются движущиеся по экрану изображения или текст, изменяющийся при щелчке мыши.

Таким образом, вы можете наблюдать JavaScript в действии, а создать подобные эффекты может любой человек, знающий этот язык. Причем без особого труда.

Следует понаблюдать за тем, что создают с помощью JavaScript другие разработчики. Это отличный способ найти свежее решение.

Прочитав эту книгу, вы не только освоите JavaScript (вместо того, чтобы просто копировать и вставлять в Web-страницу готовые сценарии), но и узнаете множество интересных идей, реализовать которые вы сможете сами - с помощью JavaScript.

Важность изучения JavaScript

Полезно изучить JavaScript хотя бы из-за его широкого распространения в сети. Огромное количество Web-страниц сделано с использованием сценариев (по самым разным причинам), и сайты без них кажутся блеклыми и скучными. Каким бы интересным не было содержание сайта, некоторые посетители сразу захотят уйти. Конечно, нет смысла оспаривать важность текстового содержания для любой Web-страницы, однако использование JavaScript не только улучшит подачу материала, но и сделает вашу страницу более запоминающейся.

Запомните: единственное, что отличает пользующиеся успехом сайты от неудачных, - их внешний вид. Чем интереснее оформлен сайт, тем он популярнее.

Остерегайтесь использования найденных в сети сценариев, предназначенных для копирования и вставки. Как правило, они не отличаются хорошим качеством и, что хуже всего, дублируются на тысячах сайтов. Изучение JavaScript избавит вас от такой практики и позволит создавать оригинальные запоминающиеся Web-страницы.

Можно найти еще множество причин для изучения и использования JavaScript. Сценарий всего в несколько строк помогает посетителям сайта отыскать страницу, соответствующую их браузеру и установкам, либо автоматически подсчитать сумму колонок в форме заказа. Подобные «мелочи» не рассчитаны на внешний эффект, но они свидетельствуют о высоком уровне профессионализма разработчика и знании им этики деловых отношений.

Ознакомившись с этой главой, вы сможете наконец создать свой первый сценарий. Вы начнете осваивать азы JavaScript на конкретных примерах. На ваше рассмотрение предлагается следующее: а основные понятия JavaScript, такие как синтаксис, разметка, комментарии и др.; а некоторые термины языка JavaScript и причины, по которым он относится к объектно-ориентированным языкам программирования; Q создание и чтение сценария JavaScript; а использование JavaScript для вывода и ввода информации.

Ввод/вывод информации с помощью JavaScript

Исследование возможностей ввода и вывода информации - это не только замечательный повод научиться создавать сценарии JavaScript, обеспечивающие двустороннюю связь с пользователем, но и прекрасное начало вашего профессионального пути.

В вычислительной технике все основано на вводе и выводе данных. Без этого ничего не происходит. Текстовый процессор не выполняет никаких действий, пока пользователь не введет какую-либо информацию (символы, набираемые на клавиатуре), и лишь затем эта информация отображается на экране, распечатывается или сохраняется на жестком диске.

Теперь вы научитесь как вводить информацию с помощью JavaScript, так и выводить ее в виде разнообразных окон сообщений. (Если вам приходилось блуждать по сети дольше нескольких минут, то вам они уже встречались.)

Объекты, методы и свойства

JavaScript - это объектно-ориентированный язык. Но что это означает? Чтобы это понять, вам следует ознакомиться с тремя терминами:

- объекты;
- методы;
- свойства.

Сначала рассмотрим их в общих чертах. Чем дальше вы продвинетесь в изучении JavaScript, тем чаще вам придется ими пользоваться, так что более близкое знакомство оставим на будущее.

Объекты

Говоря простым языком, *объект* (object) - это какой-либо предмет. Подобно тому, как в реальном мире все одушевленные и неодушевленные предметы являются объектами (машины, собаки и пр.), объектами считаются и составляющие компьютерного мира.

Что касается JavaScript, его объекты находятся внутри браузера. Это, в частности, окно браузера, формы и их части, например кнопки и текстовые окна.

В JavaScript также имеется собственная группа встроенных объектов, к которым относятся массивы, данные и т.д. Сейчас вам не обязательно фиксировать на этом внимание, поскольку все эти объекты будут рассмотрены позже. Пока вы должны усвоить лишь необходимые определения.

Именно благодаря наличию объектов язык JavaScript считается объектно-ориентированным. Язык организован вокруг объектов, а не действий, или, иначе говоря, ориентирован на данные, а не на логику. При объектно-ориентированном программировании первоочередное внимание уделяется объектам, с которыми производятся некоторые манипуляции, а не логическим правилам, необходимым для таких манипуляций. Преимуществом такого подхода является не только облегчение программирования (или написания сценария), но и в то, что каждое действие можно выполнить разными способами.

Методы

Метод (method) - это действия, которые может выполнять объект. В реальном мире у объектов тоже имеются какие-либо методы. Машины ездят, собаки лают, доллар покупается и т.д. В нашем случае alert () является методом объекта Window, то есть объект Window может выдавать пользователю какое-либо предупреждение в окне сообщений. Примерами других методов являются открытие и закрытие окон, нажатие кнопок. Здесь речь идет о трех методах: open (), close () и click (). Обратите внимание на круглые скобки. Они означают, что методы, в отличие от свойств, используются.

Свойства

У всех объектов имеются *свойства* (properties). Если вы и далее будете следовать аналогии с объектами реального мира, то обнаружите, что все предметы обладают какими-то свойствами: у машин есть колеса, а у собаки - шерсть. Что касается JavaScript, то у такого объекта, как браузер, имеется название и номер версии.

2.2. Управление данными с помощью переменных

В предыдущем разделе был анализирован использование JavaScript для отображения на экране окон сообщений трех типов, служащих для ввода-вывода информации. Однако примеры сценариев, которые вы видели, не отличались гибкостью. Вы всего лишь дополняли сценарий сообщениями, которые хотели вывести на экран. Любой текст, помещенный в окне запросов, исчезал после щелчка по кнопке выбора. В данной главе вы узнаете, как изменить эту ситуацию. Здесь рассмотрено использование значений и переменных в JavaScript.

Значения в языке JavaScript

Самое ценное в нашем мире — это информация. Каждый фрагмент информации в JavaScript рассматривается как *значение* (value). Так как информация может быть очень разнообразной, имеются различные категории значений. Простейшие типы данных в JavaScript принято называть *основными типами* (primitive types).

Три основных типа данных:

- строка (string);
- число (number);
- булево выражение (boolean).

Строки

Возможно, самым распространенным типом данных является строка. *Строка* -связанный набор символов, включающий в себя буквы, знаки препинания и цифры. В JavaScript строки чаще всего представляют собой какой-либо текст:

Здравствуйте и добро пожаловать!

Кто вы?

Мой рост - шесть футов.

Строки, вставляемые в сценарий JavaScript, заключаются в двойные или одинарные кавычки, например:

"Здравствуйте и добро пожаловать!"

Причина использования двух типов кавычек заключается в том, что двойные кавычки (") могут содержаться внутри строки, заключенной в одинарные кавычки (') , и наоборот. Например:

"J'm 6 feet tall"

'"Кто вы?" - спросил он.'

Строка может и вовсе не содержать символов. В этом случае она называется пустой строкой и обозначается пустыми кавычками:

Числа

JavaScript воспринимает два типа чисел. Это *целые числа* (integer) и *числа с плавающей точкой* (floating-point number).

Целые числа

Включают в себя положительные целые числа, например *1,2,3,* отрицательные целые числа, например -1, -2, -3 и нуль - 0.

Хотя большинство чисел, используемых в JavaScript, записываются в десятеричной системе счисления, могут применяться также восьмеричная и шестнадца-теричная системы.

Числа с плавающей точкой

Числа с плавающей точкой представляют собой числа с дробной десятичной частью:

3.14159265358979323846264338327 95

Либо это числа, выраженные в экспоненциальном виде:

3.76e2

При экспоненциальной записи числа символ <se» в верхнем или нижнем регистре означает «10 в степени»-.

Число, начинающееся с нуля и содержащее десятичную точку, считается числом с плавающей точкой.

Число, начинающееся с нескольких нулей и содержащее десятичную точку (например, 005.5) воспринимается как ошибка.

В табл. 2.1 приводятся примеры чисел, использующихся в JavaScript, чтобы вам проще было освоиться с ними.

Большие и маленькие числа

Таблица 2.1. Примеры чисел, использующихся в JavaScript

Число	**Описание**	**Десятичный эквивалент**
91	Целое число	91
4.56e2	Число с плавающей точкой	456
0.001	Число с плавающей точкой	0.001
00.001	Ошибка	
0.001	Четыре равных числа	0.001
.001	с плавающей точкой	
1e-3		
1.0e-3		

Числа, используемые в JavaScript, могут быть как очень большими, так и очень маленькими. Под очень большими подразумеваются величины до 10^{308} (единица с тремястами восемью нулями), а под очень маленькими - Ю~^{308} (нуль целых с тремястами семью нулями и единицей после запятой).

Булевы выражения

Булевы выражения отличаются от строк и чисел тем, что могут принимать лишь два значения: true (истина) и false (ложь).

Булевыми эти выражения названы в честь английского математика Джорджа Буля (1815-1864).

Ниже помещены примеры булевых выражений:

Собака лает = true

У собаки пять ног = false

К булевым выражениям вернемся в главе 6, где будут рассматриваться выражения и условия. Пока вам нужно просто знать, что такой тип данных существует и может использоваться в JavaScript.

Особые типы данных: числа, неопределенные и неопределяемые выражения.

Помимо рассмотренных типов данных, встречаются еще несколько основных, менее очевидных типов. Далее приводятся четыре числовых значения:

- положительная бесконечность;
- отрицательная бесконечность;
- положительный и отрицательный нуль;
- несуществующее число (not a number - NaN) .

Имеются еще два специальных типа данных:

- неопределенный (null);
- неопределяемый (undefined).

Что касается первых четырех типов, то вам не придется работать с ними (не будете же вы вводить в программу бесконечное число). Под несуществующим числом понимается результат бессмысленной математической операции (например, деления на нуль). Бесконечностью считается значение, превышающее 10^{308} (например, результат возведения 10^{300} в квадрат), и оно также не имеет для вас особого значения (если только не свидетельствует об ошибке).

Неопределенный тип может иметь только одно значение - null. В данном случае это свидетельствует о полном отсутствии полезной информации или каких-либо данных.

Понять, что такое неопределяемый тип данных, несколько сложнее. Здесь тоже возможно только одно значение (undefined), являющееся чаще всего тем же, что и null. В наиболее экстремальных случаях undefined становится плохим признаком (особенно если выдается в виде результата посетителю страницы) и свидетельствует о недочетах вашего сценария.

Переменные в языке JavaScript

Теперь, когда вам известно об используемых в языке JavaScript значениях, вы готовы к тому, чтобы войти в мир *переменных* (variable) JavaScript.

Переменные имеют огромное значение не только в JavaScript, но и во всех языках программирования. Без их использования трудно обойтись, а с их помощью вы сможете управлять всеми типами данных.

Переменная - это имя, присваиваемое ячейке памяти компьютера, которая хранит определенные данные во время исполнения сценария JavaScript. Возможно, это определение кажется несколько сложным, но на самом деле пользоваться переменными очень просто.

Создание переменных

В языке JavaScript переменные создаются довольно легко. Давайте вернемся к стандартному шаблону HTML, чтобы сразу ввести вас в курс дела:

```
<html>
<head>
<Title>Простая страница</Title>
<script language="JavaScript">
<!-- Маскируемся!
// Снимаем маскировку. -->
</script>
</head>
<body>
</body> </html>
```

Прежде всего нужно создать переменную. Это можно сделать двумя путями: объявив ее заранее или создав «на лету». Сначала будут рассматриваться переменные, объявляемые заранее. Позже в этой главе рассказывается, каким образом они создаются «на лету».

Пример. Чтобы объявить (создать) переменную, в языке JavaScript используется оператор var, вслед за которым указывается имя, которое вы хотите присвоить переменной. В данном примере объявляется переменная под названием msg:

```
<html>
<head>
```

```
 <title>Простая страница</title>
<script language="JavaScript">
<!-- Маскируемся!
var msg;
// Снимаем маскировку. -->
</script>
</head>
<body>
</body> </html>
```

Вот что следует знать об именах переменных:

- в именах переменных можно использовать символы нижнего и верхнего регистра либо сочетание того и другого;
- до объявления переменной ее значением считается undefined;
- имя переменной не может начинаться с цифры;
- в именах переменных недопустимы пробелы; если необходим разделитель, используется символ подчеркивания (_);
- в именах переменных следует избегать символа доллара ($) поскольку он не воспринимается, браузерами Internet Explorer 3.02 (поддерживающим JScript 1.0) и Netscape Navigator 2.02;
- следует избегать использования имен переменных, отличающихся только символами верхнего и нижнего регистра (например, msg и MsG), поскольку JScript 1.0 не сумеет их различить.

Рассмотрим следующие примеры корректных имен переменных:

msg

Hello_all Msgl Msg_l

Приведенные ниже имена переменных либо являются недопустимыми, либо их следует избегать:

- Imsg - начинается с цифры;
- helloall- содержит пробел;

- var- зарезервированное слово JavaScript;
- dollar$ - в имени содержится символ $;
- msg и Msg - не используйте такие имена в одном сценарии, поскольку различаются они только регистром.

Пример. Если вы хотите присвоить значение только что объявленной переменной, это можно сделать в той же строке:

```
<html>
<head>
<title> Простая страница </title>
<script language="JavaScript">
<!-- Маскируемся!
var msg = "Добро пожаловать в мир переменных JavaScript!";
// Снимаем маскировку. -->
</script>
</head>
<body>
</body> </html>
```

2.3. Массивы JavaScript и Escape-последовательности

В предыдущей главе вы научились пользоваться переменными. В этой главе вы узнаете о новой разновидности переменных, которые способны хранить в себе несколько значений, - о массивах. Здесь рассказывается:

- как пользоваться массивами;
- как создаются массивы;
- как создаются элементы массива;
- как пользоваться знаками переключения кода.

Как пользоваться массивами

Массив (array) - мощное средство программирования для любого языка, в том числе в JavaScript. Массив позволяет сохранять несколько независимых значений в одной переменной. Обычно эти значения как-то связаны (например, названия дней недели). Преимущество массивов заключается в том, что правильное их использование значительно упрощает код и помогает избежать создания множества переменных с похожими именами.

Итак, посмотрим, как следует создавать массивы и пользоваться ими.

Пример. В первую очередь вам нужно создать массив:

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
<!-- Маскируемся!
var days_of_week = new Array(7);
// Снимаем маскировку. -->
</script>
</head>
<body>
</body> </html>
```

Эта простая строка сценария приводит к нескольким различным последствиям:

- создается переменная days_pf_week;
- с помощью new Array () указывает, что новая переменная является массивом; а определяется размер массива (в данном случае - 7).

Таким образом, созданы семь пустых ячеек, или *элементов,* которым нужно присвоить некоторые значения.

Ввести значения достаточно просто, только не забывайте отслеживать, какое значение присваивается каждому элементу.

days_of_week[x] = *значение;*

В данном случае x означает номер элемента.

Прежде всего обратите внимание, как JavaScript нумерует элементы. Он рассматривает первый элемент массива не как первый (с номером 1), а как нулевой (с номером 0). То есть в данном примере дням недели будут соответствовать порядковые номера от 0 до 6, а не от 1 до 7.

Пример. Итак, чтобы присвоить соответствующее значение первому элементу массива (в данном случае это понедельник), выполняются следующие действия:

```
<html>
<head>
<title>Простая страница</title>
<script language=" JavaScript">
< ! - - Маскируемся!
var days_of_week = new Array (7), •
days_of_week[0] = "Понедельник";
// Снимаем маскировку. -->
</script>
</head>
<body>
</body> </html>
```

Пример. Следуя этому образцу, вы можете заполнить массив соответствующими значениями:

```
<html>
<head>
<title> Простая страница </title>
<script language="JavaScript">
<!-- Маскируемся!
var days_of_week = new Array(7);
```

```
                "Поне
                                    days_of_week[0]
дельник";
                                    days_of_week[1]
"Вторник";
                                    days_of_week[2]
days_of_week[3]
days_of_week[4]
days_of_week[5]
days_of_week[6]
// Снимаем маскировку. -->
</script>
</head>
<body>
</body> </html>
```

Теперь вы можете извлекать из массива значения элементов. Как ранее элементам присваивались значения посредством имени переменной (в данном случае days_of _week) и указанного в квадратных скобках номера элемента (например, days_of_week[2]), так теперь этот формат используется для извлечения значений из массива.

Пример. Таким образом, чтобы вывести на экран значение третьего элемента в окне предупредительных сообщений, выполняются следующие действия:

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
<!-- Маскируемся!
var days_of_week = new Array(7),•
days_of_week[0] = "Понедельник";
days_of_week[1] = "Вторник";
```

```
 days_of_week[2] = "Среда";
days_of_week[3] = "Четверг";
days_of_week[4] = "Пятница";
days_of_week[5] = "Суббота";
days_of_week[6] = "Воскресенье";
alert(days_of_week[2]);
// Снимаем маскировку. -->
</script>
</head>
<body>
</body>
</html>
```

Не забывайте, что счет элементов массива начинается с нуля.

Пример. Здесь представлен еще один способ, которым достигается тот же результат:

```
<html>
<head>
<title> Простая страница </title>
<script language="JavaScript">
<!-- Маскируемся!
var days_of_week = new Array(7);
days_of_week[0] = "Понедельник";
days_of_week[1] = "Вторник";
days_of_week[2] = "Среда";
days_of_week[3] = "Четверг";
days_of_week[4]   =   "Пятница";
days_of_week[5] = "Суббота";
days_of_week[6] = "Воскресенье";
var x = 2;
alert(days_of_week[x]);
```

```
 // Снимаем маскировку. -->
</script>
</head>
<body>
</body>
</html>
```

В данном случае создается переменная x и ей присваивается значение 2. Затем имя переменной x указывается в квадратных скобках, в результате чего из массива извлекается элемент с номером 2.

2.4. Выражения, условия, операции, строки и числа

Из предыдущей главы вы узнали, как в JavaScript пользоваться переменными и массивами для сохранения информации. В этой главе вы закрепите усвоенный материал и научитесь работать с этой информацией. Здесь рассматриваются следующие понятия:

- выражения;
- условия;
- различные операции и способы их использования;
- преобразование строки в число и наоборот.

Что такое выражения и условия

При изучении любого языка программирования (и JavaScript в том числе) время от времени приходится отвлекаться на попутные замечания. Давайте на минуту оторвемся от упражнений и рассмотрим новые термины.

Выражения и условия

С помощью переменных и массивов вы сохраняете в памяти компьютера любую нужную вам информацию в различных видах. Однако впоследствии вы можете сделать с этой информацией нечто большее, чем просто вывести ее на экран или поместить в окно предупредительных сообщений. Вы можете изменить ее,

управлять ею или подвергнуть ее проверке. Для этих целей и предназначены выражения и условия.

Выражения (expressions) используются для комбинации двух или более значений, в результате чего получается третье, новое значение. Примером может служить сумма в следующем выражении:

1 + 2 = 3

Вы скомбинировали два значения (1 и 2), чтобы получить третье (3). Вот еще один пример:

3 + 3 - 1 = 5

Здесь комбинация трех значений (3,3 и 1) приводит к появлению нового значения (5).

Условия (conditions) позволяют сравнивать величины и определять логическое значение - true или false. Ниже приводится пример условия:

Лимоны желтые? Да.

Или другой пример:

3 + 3 = 6? Да.

Заметьте, что на оба этих вопроса можно дать только два ответа - да или нет (хотя на вопрос о лимонах можно ответить «наверное»). В языке JavaScript (и в других компьютерных языках) использование условий подразумевает только два результата - да или нет.

Изучение JavaScript полезно еще и тем, что в процессе учебы вы получаете представление о других языках программирования. К тому же после освоения одного языка легче овладевать остальными.

Знакомство с операциями

В выражениях и условиях данные комбинируются с помощью *операций* (operator). Если манипуляции осуществляются с какой-либо одной величиной, то такая операция именуется *операцией с одним операндом* (unary operator). Если таких величин две, то операция называется *операцией с двумя операндами* (binary operator), а если три - *операцией с тремя операндами* (ternary operator).

Давайте рассмотрим наиболее часто используемые операции.

Арифметические операции

Арифметические операции - это всем известные математические действия:

а сложение (+);

1 + 3 = 4 а вычитание (-);

2 - 1 = 4 Q деление (/);

4 / 2 = 2 а умножение (*);

2 * 2 = 4 а остаток от деления (%).

9 % 5 = 4

Это были операции с двумя операндами.

Единственная арифметическая операция, нуждающаяся в пояснении, - остаток от деления. То есть 9 делится на 5 с остатком 4. Иногда эту операцию также называют взятием по модулю. При использовании целых чисел результат также будет представлять собой целое число. Но если речь идет о действительных числах (с плавающей точкой, не целых), в результате получится действительное число. Например:

5.5 % 2.2 = 1.1

Будьте внимательны при использовании этих операций, поскольку при некорректном их выполнении они могут привести к результатам NaN (несуществующее число) или Infinity (бесконечность). Примером такой проблемной операции является деление на нуль.

Примеры сценариев JavaScript

Далее описывается несколько примеров использования арифметических операций. Здесь приводится только содержимое блока SCRIPT. Все примеры созданы на основе шаблона.

Пример. Операция сложения (+) -.

```
<script language="JavaScript">
<!-- Маскируемся!
var a = 6, b = 4;
alert(a + b);
```

// Снимаем маскировку. -->

</script>

Пример. Операция вычитания (-) -.

<script language="JavaScript">

<!-- Маскируемся!

var a = 6, b = 4;

alert(a - b);

// Снимаем маскировку. -->

</script>

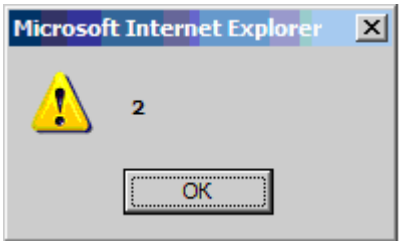


Рис. 2.4.1. Использование операции вычитания

Пример. Операция деления (/) -.

<script language="JavaScript"> <!-- Маскируемся! var a = 6, b = 3; alert(a / b);

Операции сравнения

Операции сравнения используются для сопоставления выражений. К ним относятся следующие операции:

меньше (<);

5<6

меньше или равно (<=);

6 <= 6 или 6 <= 7

а больше (>);

7 > 4

больше или равно (>=);

5 >= 5 или 5 >= 4

равно (==);

```
5 == 5
```

не равно (!=).

```
5 != 3
```

Это были операции с двумя операндами.

В этих примерах сравниваются числовые данные, но те же самые операции могут выполняться и со строками (об этом речь пойдет позже). Единственное условие состоит в том, что нужно сопоставить величины, относящиеся к одному и тому же типу. В противном случае JavaScript попытается перевести данные из одного типа в другой, что не всегда удается. Чтобы избежать ошибок, сравнивайте данные только одного типа.

Логические операции

Принцип действия *логических операций* не так очевиден. Их функции станут понятнее, когда вы начнете использовать их с операторами, например с оператором if.

Далее перечислены три логические операции:

- логическое И (and);
- логическое ИЛИ (or);
- логическое НЕ (not).

Логическое И и логическое ИЛИ - операции с двумя операндами, а логическое НЕ - операция с одним операндом. Они позволяют свести воедино результаты сравнений нескольких переменных.

Логическое И (&&) означает, что обе части выражения должны быть истинны. В качестве примера из жизни можно взять мысли водителя перед нажатием на педаль тормоза: машина едет слишком быстро && нужно затормозить.

Логическое ИЛИ (I I) означает, что, по крайней мере, одна часть выражения должна быть истинной. Снова представим себе мысли водителя перед тем, как он включает фары: темнеет I I плохая видимость.

Логическое НЕ изменяет значение истина/ложь на обратное. Например, фары включают, когда! светлеет (то есть когда темнеет).

Операции с одним операндом

Как видно из названия, эти операции осуществляются с одной величиной. К ним относятся:

префиксное и постфиксное возрастание (increment);

++

префиксное и постфиксное уменьшение (decrement);

унарный плюс;

+ унарный минус.

Унарный минус изменяет знак выражения на противоположный. Из всех четырех операций это самая простая. Кроме того, унарный плюс используется не для смены знака, а для преобразования операнда в число (например, если это была строка).

Префиксное/постфиксное возрастание/уменьшение позволяет увеличить или уменьшить значение переменной на единицу. Однако результаты выполнения этих операций зависят от того, префиксная это операция или постфиксная.

Результат использования префиксной операции проще предугадать: если a = 5, то ++a + 2 = 8, потому что значение переменной *a* было увеличено на единицу, прежде чем к нему прибавили число 2. В то же время

--a + 2 = 6,

потому что значение переменной a было уменьшено на единицу, прежде чем к нему прибавили число 2.

Постфиксные операции действуют иначе, поскольку возрастание или убывание производится только после использования старого значения в выражении. Таким образом, если a = 5, то a++ + 2 = 7, потому что увеличение на

единицу будет выполнено после вычисления значения выражения. Это справедливо и для следующего выражения:

a-- + 2 = 6,

потому что уменьшение переменной, а на единицу будет выполнено после использования в выражении прежнего значения.

2.5. Создание сценариев с помощью функций и событий

До сих пор ваши сценарии JavaScript состояли из операторов, выполняемых по следовательно - от первого до завершающего. Это оптимальное решение для простых сценариев, но в большинстве случаев при загрузке страницы выполнять сценарий целиком не нужно. Например, вы хотите, чтобы несколько операторов были задействованы вначале, а остальные - в какой-либо другой момент. Именно тогда вам потребуются функции.

Что такое функция

Функция (function) - это группа операторов, предназначенных для определенной цели и объединенных под общим именем. Функция имеет следующий вид:

```
function имяфункции()
{
операторы; }
```

В начале функции помещается слово function, за которым указывается ее имя (например, yourMessage). После имени ставятся круглые скобки (your-Message ()). Их отсутствие приводит к ошибке.

После круглых скобок идут открывающая и закрывающая фигурные скобки, между которыми помещаются операторы.

У каждой функции должно быть имя, причем имена функций, используемых на одной странице, не должны повторяться. Функция запускается (активизируется или вызывается) с Web-страницы.

Ваша первая функция

Пример. Теория может казаться вам слишком сложной, пока вы не перейдете к практике. Давайте сразу создадим первую функцию. Выполните следующие действия:

1. Откройте HTML-шаблон в текстовом редакторе.

2. Напечатайте в блоке SCRIPT слово function и дайте функции имя, например yourMessage (не забудьте о круглых скобках в конце строки):

```
<script language=" JavaScript ">
<        ! - - Маскируемся !
function yourMessage ()
// Снимаем маскировку. --> </script>
```

3. Затем добавьте пару фигурных скобок:

```
<script language=" JavaScript ">
<        ! - - Маскируемся !
function yourMessage ()
// Снимаем маскировку. --> </script>
```

Помните, что в начале ставится открывающая фигурная скобка ({), а в конце - закрывающая (}). Распространенная ошибка новичков заключается в том, что они питают скобки местами, в результате чего сценарий не работает.

4. Далее добавьте простой оператор:

```
<script language="JavaScript">
<!-- Маскируемся!
function yourMessage()
{
alert("Ваша первая функция!");
}
```

```
// Снимаем маскировку. -->
</script>
```

5. Сохраните результат, откройте его в браузере и посмотрите, что получилось.

Конечно, сначала вы ничего не увидите. Однако из этого вовсе не следует, что с вашим сценарием что-то не так. Причина в том, что вы еще не *вызвали* функцию. Это обязательно нужно сделать, потому что в отличие от простых операторов в блоке SCRIPT (запускаемых сразу) функции автоматически запускаться на выполнение не могут. Вам придется кое-что добавить в сценарий.

Функции в языке JavaScript могут вызываться несколькими способами. Самый простой, хотя и наименее эффективный - указание имени функции непосредственно в блоке SCRIPT:

```
<script language="JavaScript"> <!-- Маскируемся!
yourMessage();
function yourMessage()
{
alert("Ваша первая функция!");
}
// Снимаем маскировку. -->
</script>
```

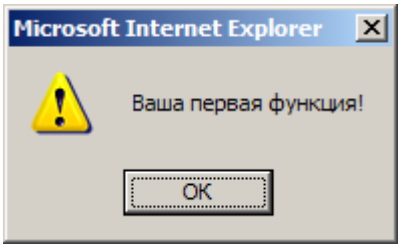


Рис. 2.5.1

Сохраните страницу еще раз и откройте ее в браузере, чтобы увидеть результат. Появление окна подтверждающих сообщений доказывает, что ваша функция была вызвана и выполнена.

Способ, с помощью которого вы вызвали функцию в этом примере, - не самый удачный. По сути, он ничем не отличается от простого перечисления операторов в блоке SCRIPT, освоенного ранее.

Предпочтительнее вызывать одну функцию из другой. Но, прежде чем вы научитесь это делать, вам предстоит узнать, каким образом выполняются функции и что такое *события* (events), поскольку именно они дают возможность управлять выполнением функций.

События

Необходимо сделать так, чтобы при вызове функций всю работу вместо вас выполняли события. JavaScript - *язык, управляемый событиями* (event-driven). То есть все происходящее в нем является результатом события или вызывает какое-либо событие. Открытие новой страницы в браузере, перемещение курсора, щелчок мыши - все это относится к событиям. В этой книге вам встретятся четыре их вида:

- onLoad;
- onclick;
- onMouseover;
- onMouseout.

Давайте коротко рассмотрим каждый из них.

Событие onLoad

Это событие происходит после загрузки чего-либо, например после открытия страницы в окне браузера. Оно считается состоявшимся только после завершения загрузки всей страницы, включая изображения.

Событие onLoad очень удобно использовать в сценарии, когда необходимо, чтобы функция выполнялась сразу после открытия страницы.

Событие onClick

Это событие происходит после щелчка мышью в определенном месте страницы. Позднее вы узнаете, что множество элементов страницы (гиперссылки, изображения, кнопки и пр.) могут реагировать на событие onClick. Его рекомендуется применять, когда вы хотите создать сценарий, взаимодействующий с пользователем.

Событие onMouseover

Это событие похоже на событие onCl ick, но происходит не после щелчка мышью, а после наведения курсора на определенный элемент страницы. Событие onMouseover можно связать практически с любым объектом Web-страницы (текстом, изображением, кнопками, гиперссылками и т.д.). Его также можно использовать, чтобы повысить уровень интерактивности сценария.

Событие onMouseout

Это событие подобно событию onMouseover, но происходит в тех случаях, когда курсор мыши отводится от объекта.

3. Разработка учебных программ по предмету "Web программирования"

3.1. Описание программ

Разработанные программные примеры предназначены студентам для выполнения практических задач по предмету "Web программирования".

Разработанные примеры с использованием JavaScript соответствует рабочей программы практических занятие. Ниже приведён список разработанных программ:

- проверка правильности электронного адреса (по заданному шаблону);
- проверка заполнения полей формы (проверка на наличие данных);
- пример формы заказа (объединение всех проверок на уровне формы);
- фоновая музыка;
- простая анимация на JavaScript;
- пример таблицы стилей (без использования JavaScript);
- применение динамических стилей, используя JavaScript;
- управление слоями, используя JavaScript;
- создание анимации с помощью DHTML;
- поисковая система (написана на JavaScript);
- дополнительные примеры.

Примеры программ разработаны в виде Web-сайт. Каждый пример оформлен в виде отдельной Web-страницы.

Для работы с программой необходимо запустит файл index.htm. После открытия запуска файла index.htm появляется следующие окно (рис.3.1.1):

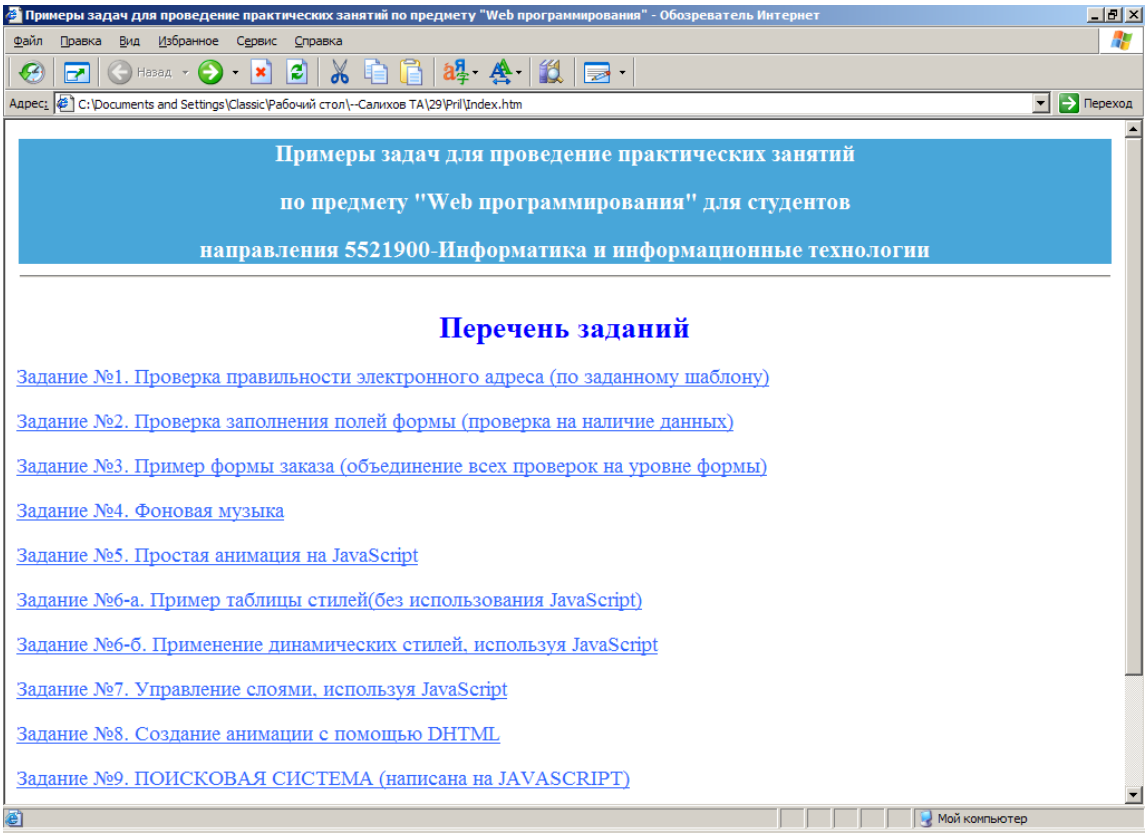


Рис.3.1.1. Главная страница

В главной странице приведен список примеров задании. С нажатием нужную ссылку открывается соответствующий пример.

Теперь подробно рассмотрим каждый из задачи.

1. Проверка правильности электронного адреса (по заданному шаблону) (рис.3.1.2)

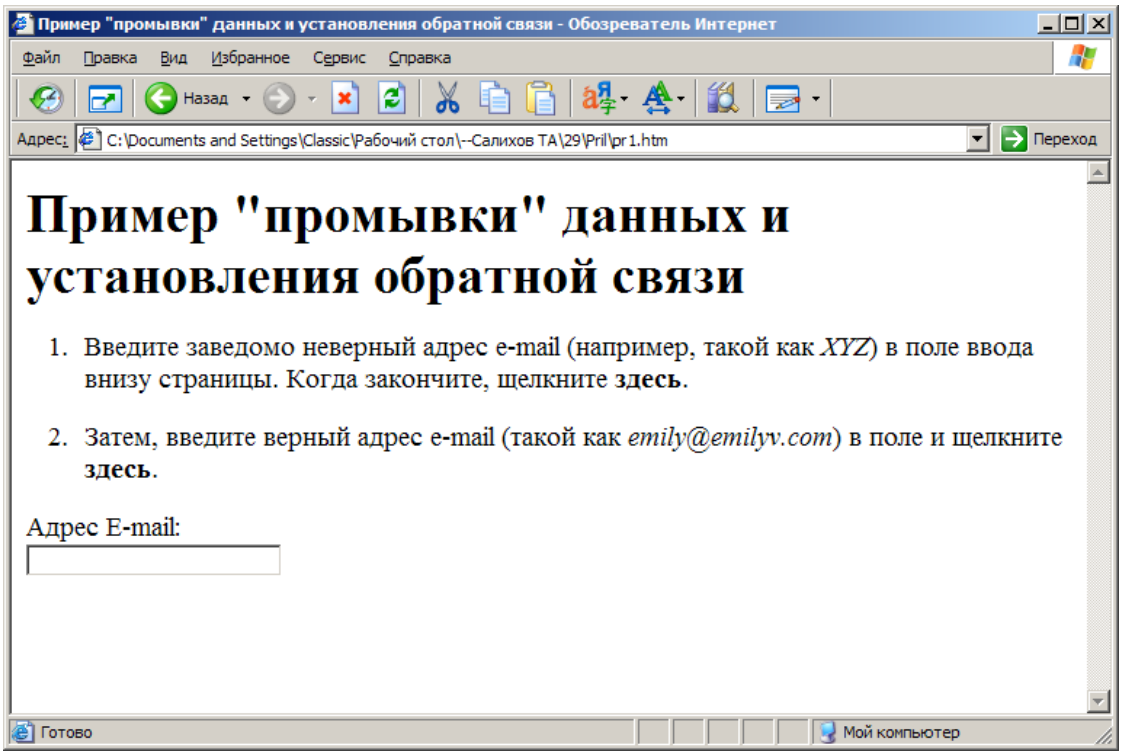


Рис.3.1.2. Проверка правильности электронного адреса (по заданному шаблону)

Чтобы проверить действие программы нужно ввести в окошку «Адрес E-mail:» неверный адрес e-mail (например, такой как *XYZ*) и щелкнут текст «**здесь**». После щелчка текста «**здесь**» на экране появляется окно сообщения:

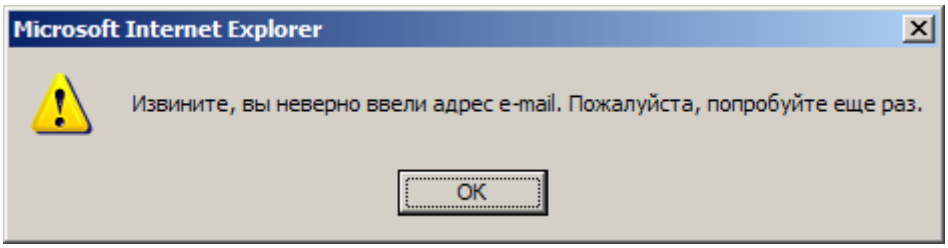


Рис. 3.1.3. Результат неправильного ввода e-mail адреса

Если вводится верный адрес e-mail (такой как *emily@emilyv.com*) в поле и щелкнут текст «**здесь**» на экране появляется следующем виде окно сообщения:

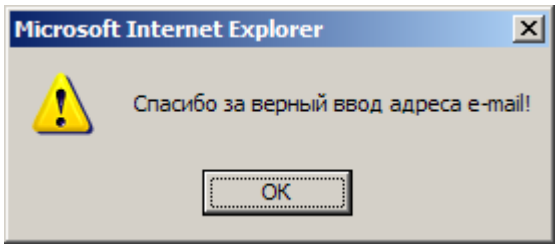


Рис. 3.1.4. Результат правильного ввода e-mail адреса

2. Проверка заполнения полей формы (проверка на наличие данных) (рис.3.1.5).

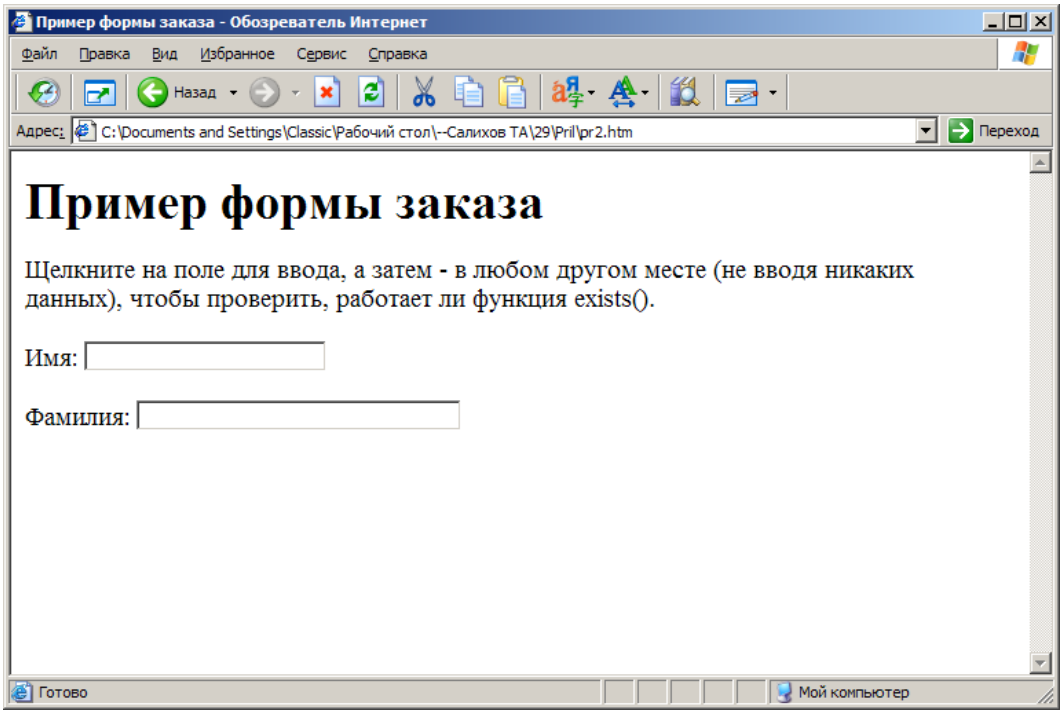
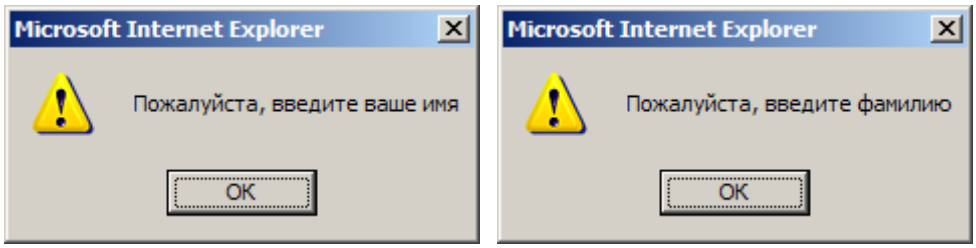


Рис.3.1.6. Проверка заполнения полей формы (проверка на наличие данных)

Чтобы увидеть результаты выполнения примера щелкнут на поле для ввода, а затем - в любом другом месте (не вводя никаких данных), чтобы проверить, работает ли функция exists(). В результате появляется следующие окно сообщение:



а) б)

Рис.3.1.7. Результаты выполнения примера «Проверка заполнения полей формы»: а) не заполнен поле «Имя», б) не заполнен поле «Фамилия»

3. Пример формы заказа (объединение всех проверок на уровне формы) (рис.3.1.7).

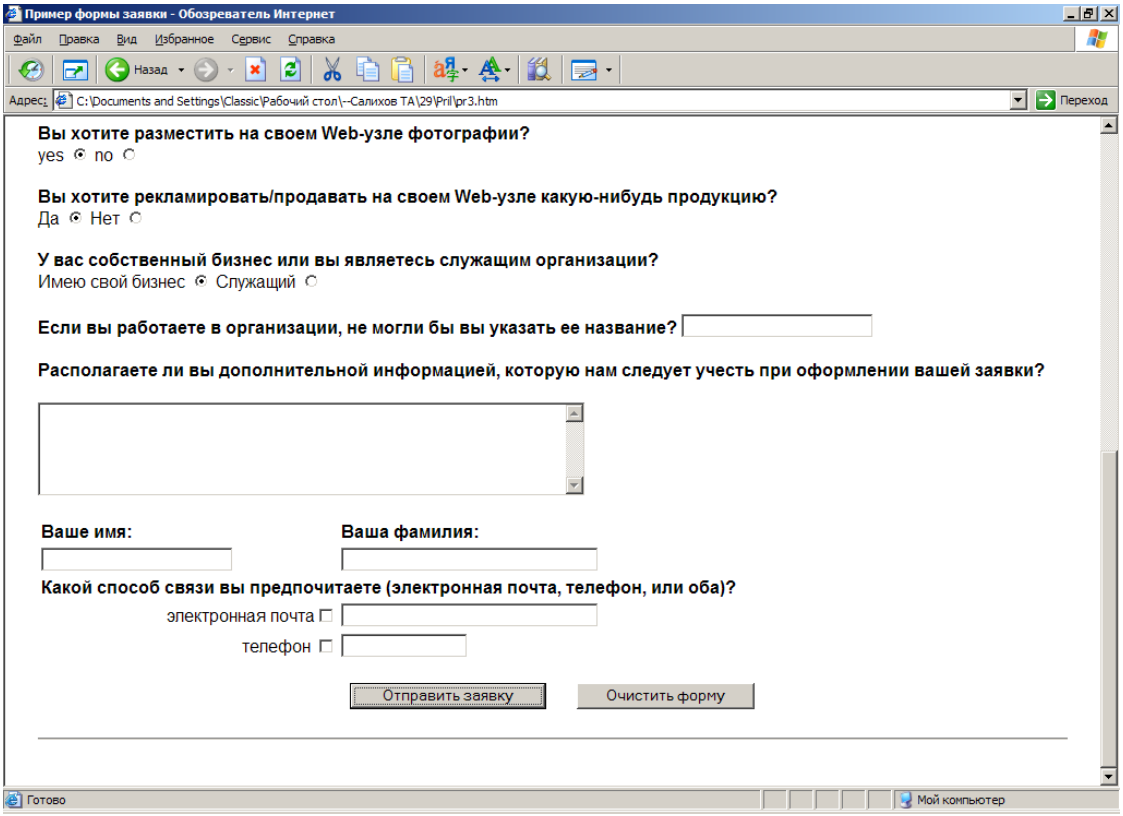


Рис. 3.1.7. Объединение всех проверок на уровне формы

Если не заполняется, какой нибуд нужное поля то с нажатиям кнопку «Отправить заявку» на экране появляется по очереди, следующие сообщении:

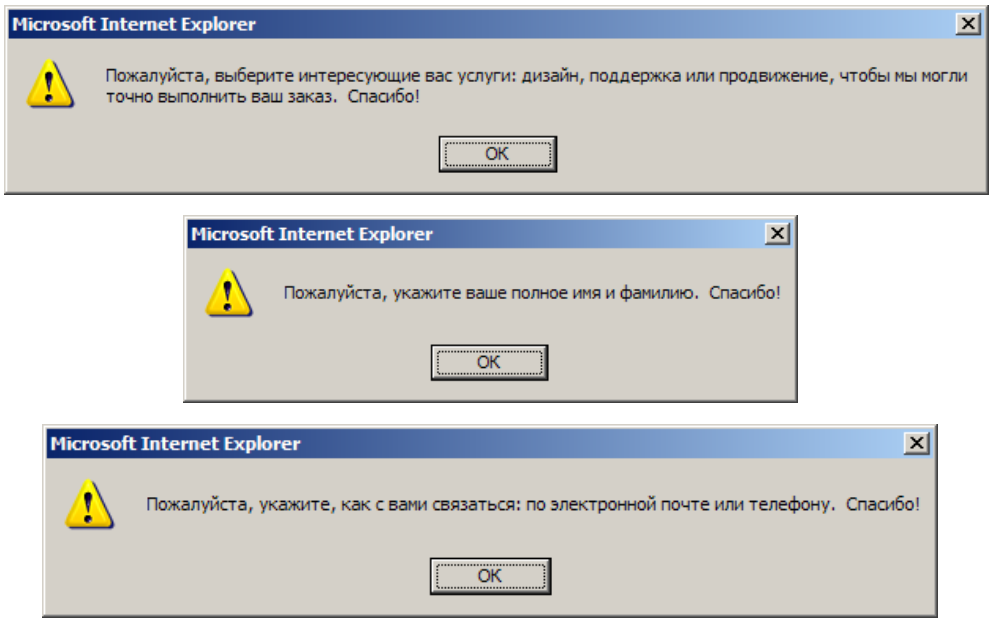


Рис. 3.1.8. Окно сообщении о незаполненных полей

4. Фоновая музыка (рис.3.1.9).

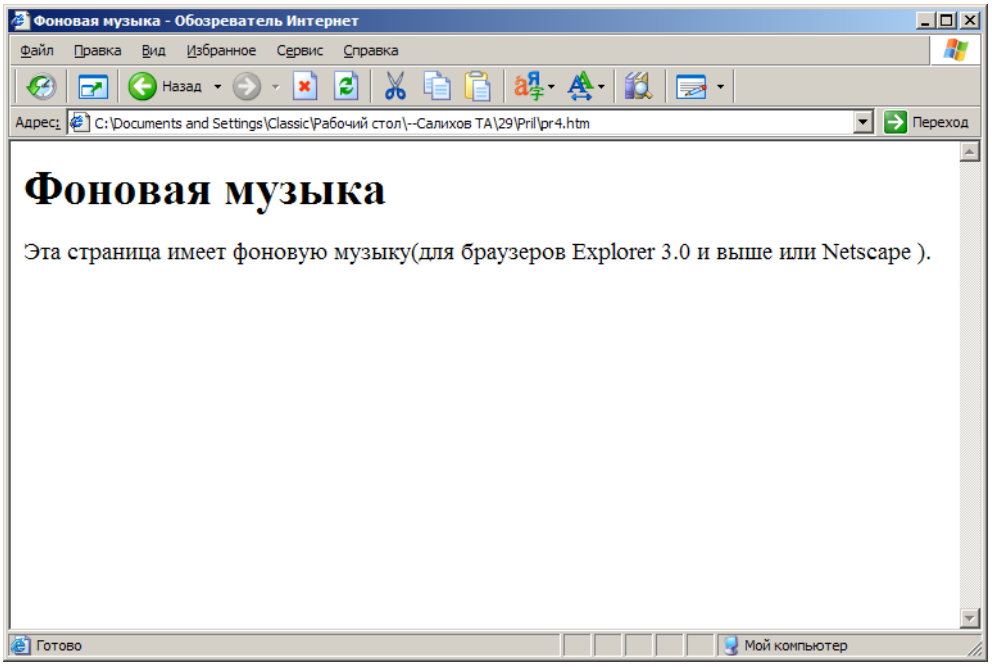


Рис.3.1.9. Окно примера «Фоновая музыка»

После открытия этой страницы будет слышно музыка, который указан в программе.

5. Простая анимация на JavaScript

В примере приведен метод создания простых анимации на JavaScript.

6. Пример таблицы стилей(без использования JavaScript) (рис.3.1.10).

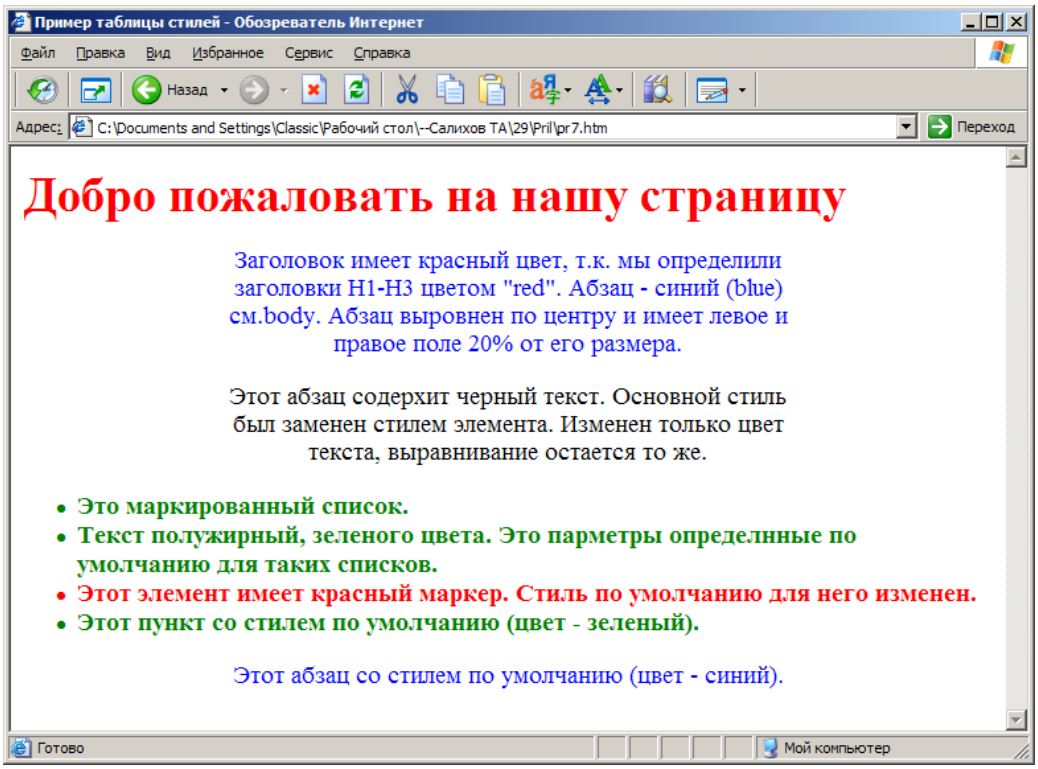


Рис.3.1.10. Пример оформления цвета текста (без использования JavaScript)

Применение динамических стилей, используя JavaScript (рис.3.1.12).

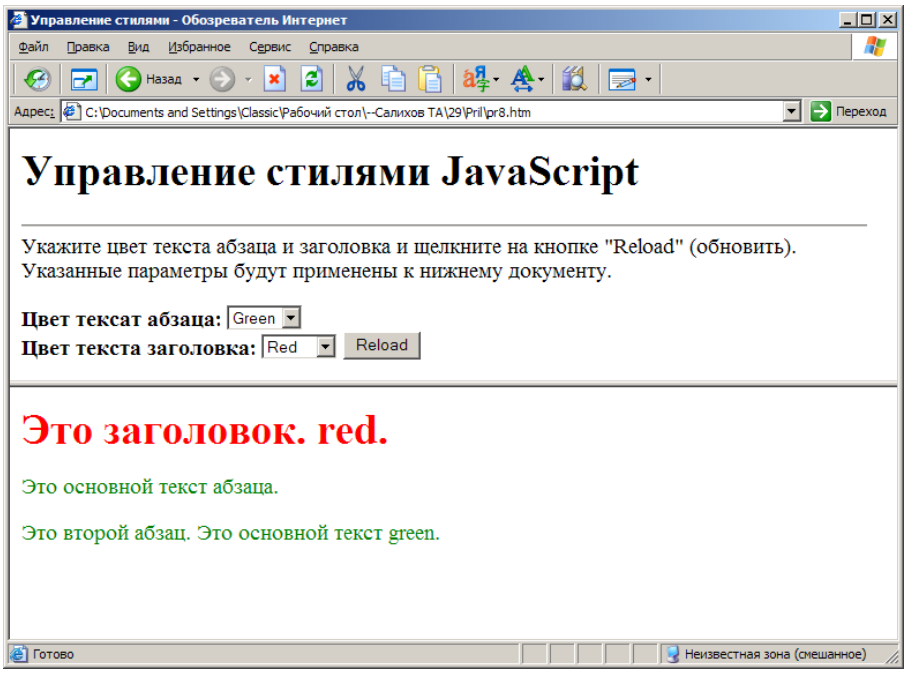


Рис.3.1.11. Применение динамических стилей, используя JavaScript

Чтобы изменить цвет текста абзаца и заголовку, необходимо указать цвет текста абзаца и заголовка после щелкнуть на кнопке "Reload" (обновить). Указанные параметры будут применены к нижнему документу.

7. Управление слоями, используя JavaScript

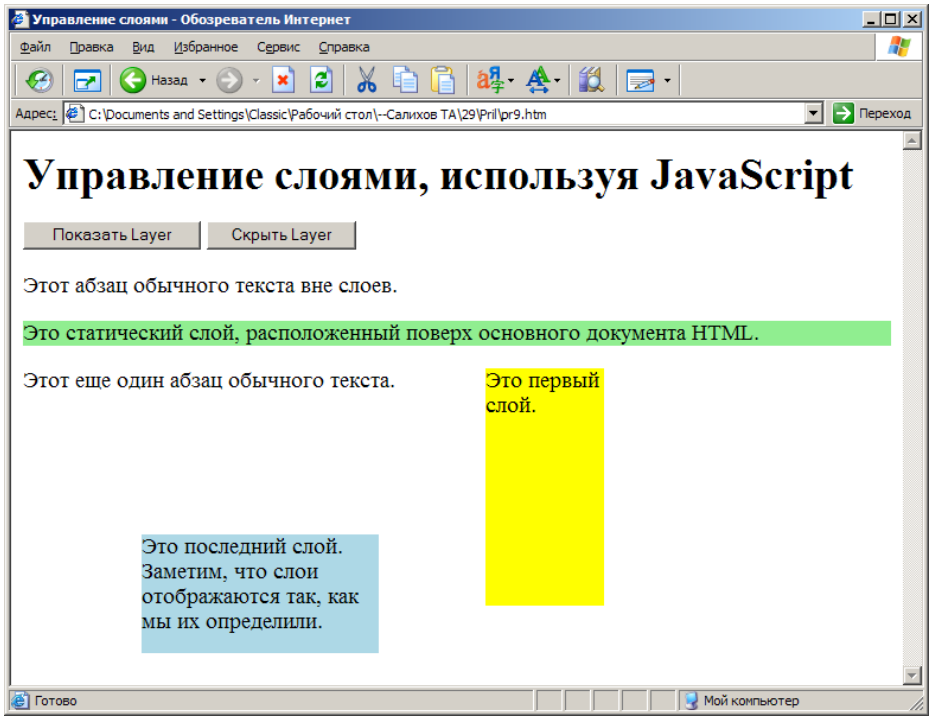


Рис. 3.1.12. Управление слоями, используя JavaScript

Если нажимается кнопка «Показать Layer» в участке рабочей области появляется новый слой (рис.3.1.13).

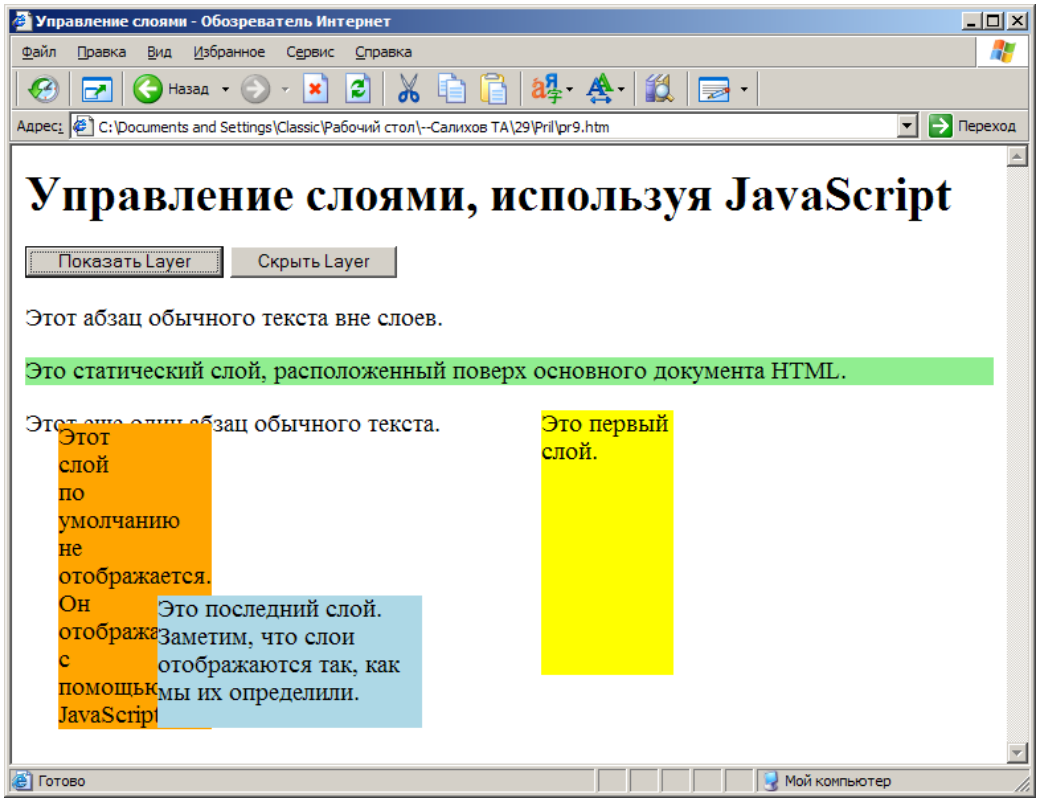


Рис.3.1.13.

Чтобы убрать этот слой необходимо нажать кнопку «Скрыть Layer».

8. Создание анимации с помощью DHTML

На этом примере показано создание анимации с помощью DHTML.

9. Поисковая система (написана на JavaScript)(рис.3.1.14).

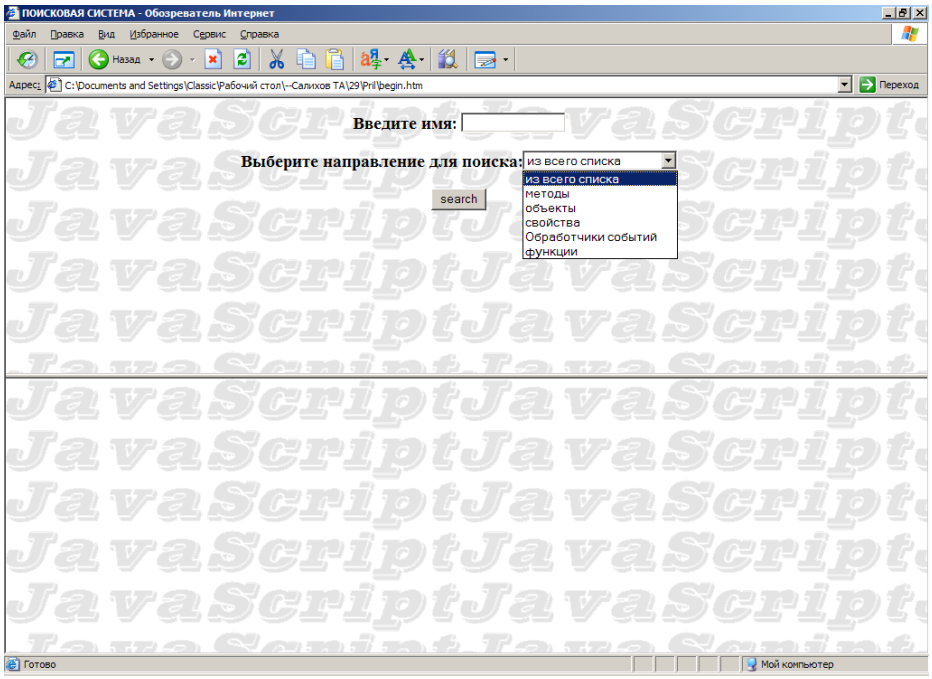


Рис. 3.1.14. Поисковая система (написана на JavaScript)

С использованием JavaScript разработано поисковая система, которое предназначена для поиска и сортировки методов, объектов, свойств, обработчика событий и функции JavaScript.

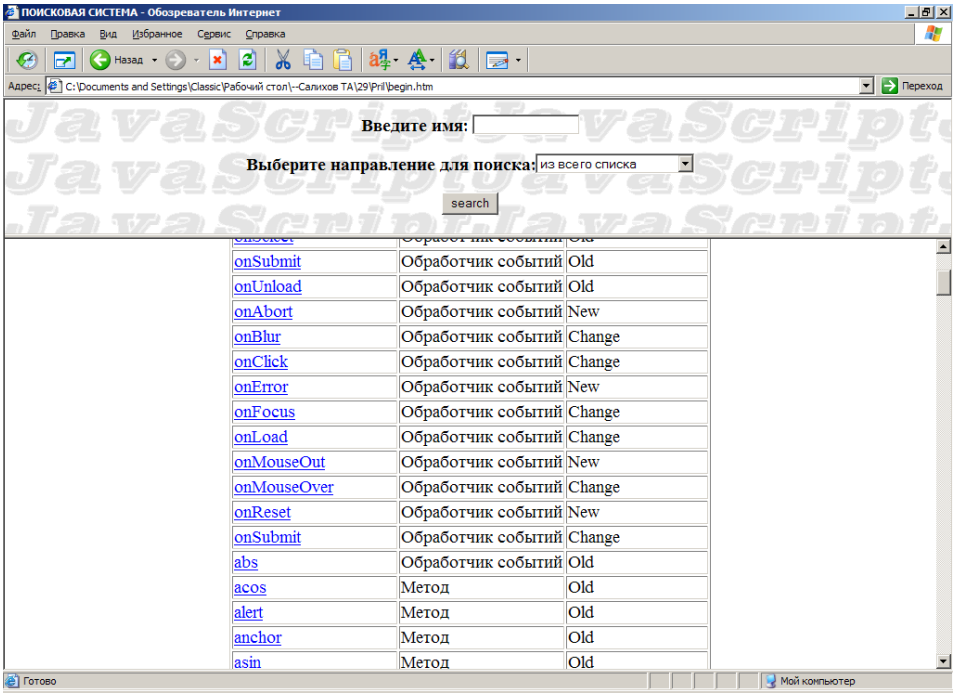


Рис.3.1.15. Результаты поиска

Кроме приведенных девяти примеров, разработаны ещё несколько дополнительных примеров (рис.3.1.16).

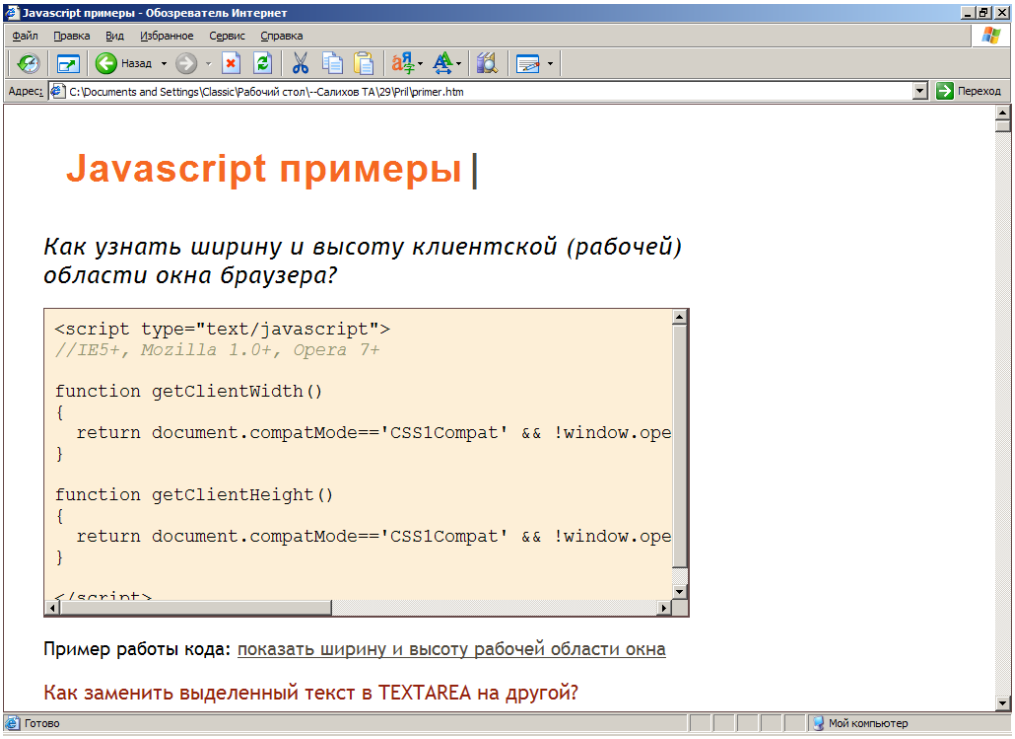


Рис.3.1.16. Дополнительные примеры

Студенты могут пользоваться исходными кодами выше перечисленных примеров. Исходные коды примеров можно открыт выбрав команду «Просмотр HTML-кода» из меню «Вид» (рис.3.1.17).

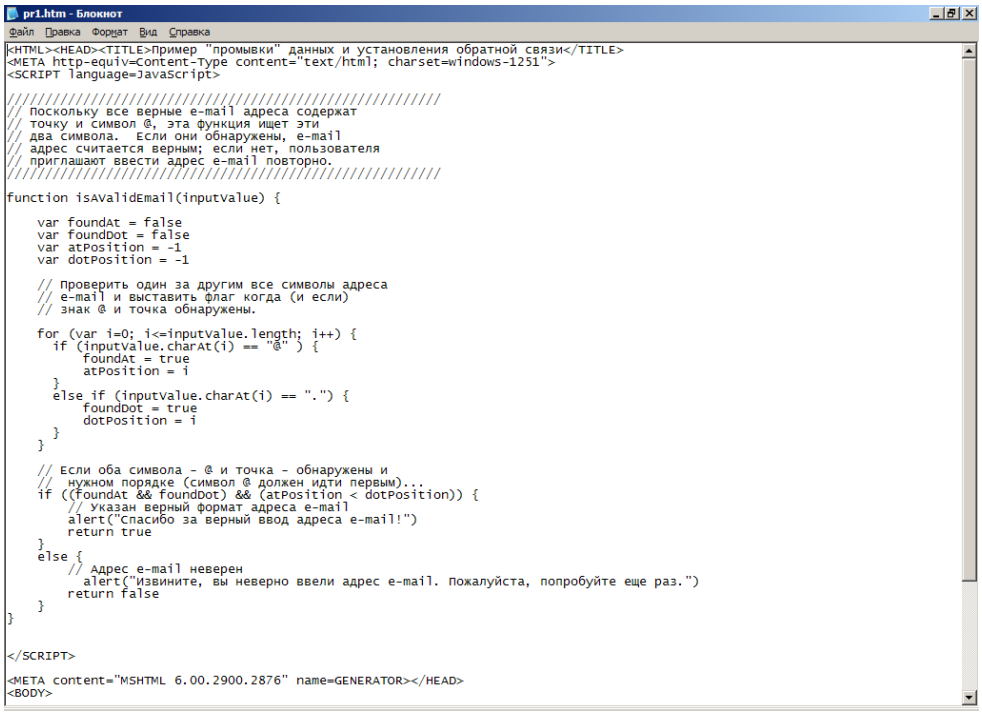


Рис.3.1.17. HTML-код примера

3.2. Руководство пользователя

Для нормальной работы разработанной программы необходимо имеет следующие аппаратно-программные средства:

1) Процессор: PENTIUM II и выше.
2) Оперативная память не менее 64 МБ.
3) Не менее 10 МБ память на винчестере.
4) Операционная система: Windows 98, NT, XP или Vista.

Для работы с примерами программ нужно открыть главное страницу (index.htm). Из предлагаемого списка примеров выбирается нужный пример.

4. Охрана труда и техника безопасности

4.1. Цель и содержание БЖД

С развитием научно-технического прогрессе немаловажную роль играет возможность безопасного исполнения людьми своих трудовых обязанностей. В связи с этим была создана и развивается наука о безопасности труда и жизнедеятельности человека.

Безопасность жизнедеятельности – это комплекс мероприятий, направленных на обеспечение безопасности человека в среде обитания, сохранение его здоровья, сохранение его здоровья, разработку методов и средств защиты путем снижения влияния вредных и опасных факторов до допустимых значений, выработку мер, но ограничению ущерба в ликвидации последствии чрезвычайных ситуаций мирного и военного времени. Охрана здоровья трудящихся, обеспечение безопасности условий труда, ликвидация профессиональных заболеваний и производственного травматизма составляет одну из главных забот человеческого общества. Обращается внимание на необходимость широкого применения прогрессивных форм научной организации труда, сведения к минимуму ручного, малоквалифицированного труда, создания обстановки, исключающей профессиональные заболевания и производственный травматизм. На рабочем месте должны быть предусмотрены меры защиты от возможного воздействия опасных и вредных факторов производства. Уровни этих факторов не должны превышать предельных значений, оговоренных правовыми, техническими и санитарно-техническими нормами. Эти нормативные документы обязывают к созданию на рабочем месте условий труда, при которых влияние опасных и вредных факторов на работающих либо устранено совсем, либо находится в допустимых пределах. Данный раздел дипломного проекта посвящен рассмотрению следующих вопросов:

- определения оптимальных условий труда инженера-программиста;
- расчет освещенности;

- обеспечение пожаробезопасности.

4.2. Характеристика условий труда программиста

Научно-технический прогресс внес серьезные изменения в условия производственной деятельности работников умственного труда. Их труд стал более интенсивным, напряженным, требующим значительных затрат умственной, эмоциональной и физической энергии. Это потребовало комплексного решения проблем эргономики, гигиены и организации труда, регламентации режимов труда и отдыха.

В настоящее время компьютерная техника широко применяется во всех областях деятельности человека. При работе с компьютером человек подвергается воздействию ряда опасных и вредных производственных факторов: Электромагнитных полей (диапазон радиочастот: ВЧ, ЧВЧ и СВЧ), инфракрасного и ионизирующего излучений, шума и вибрации, статического электричества.

Работа с компьютером характеризуется значительным умственным напряжением и нервно-эмоциональной нагрузкой операторов, высокой напряженностью зрительной и достаточно большой нагрузкой на мышцы рук при работе с клавиатурой ЭВМ. Большое значение имеет рациональная конструкция и расположение элементов рабочего места, что важно для поддержания оптимальной рабочей позы человека-оператора.

В процессе работы с компьютером необходимо соблюдать правильный режим труда и отдыха. В противном случае у персонала отмечается значительное напряжение зрительного аппарата с появлением жалоб на неудовлетворенность работой, головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

4.3. Эргономические требования к рабочему месту

Проектирование рабочих мест, снабженных видеотерминалами, относится к числу важных проблем эргономического проектирования в области вычислительной техники.

Рабочее место и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия: оптимальное размещение оборудования, входящего в состав рабочего места и достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения.

В помещениях, где находится компьютер, необходимо обеспечить следующие величины коэффициента отражения: для потолка – 60 ÷ 70 %, для стен – 40 ÷ 50%, для пола – 30%, для других поверхностей и рабочей мебели – 30 ÷ 40%.

Рабочее место программиста и оператора должно быть организовано таким образом, чтобы:

- высота стола с клавиатурой составляла 62-88 см над уровнем стола; а высота экрана (над полом) – 90 ÷ 128 % см;
- расстояние от экрана до края стола – 40 – 115 см;
- наклон экрана от – 15 до + 20^0 по отношению к нормальному его положению;
- положение спинки кресла оператора обеспечивало наклон тела назад 97: 121^0.

Клавиатуру следует делать отдельной от экрана и подвижной. Усилие нажима на клавиши должно лежать в пределах 0,25:1,5 Н, а ход клавишей – 1:5 мм.

Существенное значение для производительной и качественной работы на компьютере имеют размеры знаков, плотность их размещения, контраст и соотношение яркостей символов и фона экрана. Если расстояние от глаз оператора до экрана дисплея составляет 60÷80см, то высота знака должна быть не менее 3 мм, оптимальное соотношение ширины и высоты знака составляет 3 : 4, а расстояние между знаками – 15 : 20 % их высоты. Соотношение яркости фона экрана и символов от 1 : 2 : 1 : 5 до 1 : 10 : 1 : 15.

Подкладка под запястья при работе с клавиатурой позволит избежать болезни кистей.

Кроме всего прочего рекомендуется иметь специальные протирочное средство для мониторов, которое не только очищает экран от грязи и пыли, но и покрывает его антистатическим слоем.

4.4. Метрологические условия и их влияние на организм человека

Состояние воздушной среды производственных помещений характеризуется степенью частоты воздуха и метрологическими условиями, под которыми понимают различные сочетания температуры, влажности и скорости движения воздуха. Длительное действия на человека неблагоприятных метрологических условий резко ухудшает его самочувствие, снижает производительности труда и часто приводит к различным заболеваниям.

Метрологические условия для рабочей зоны устанавливается в соответствии с требованиями санитарных норм.

В качестве оптимальных мете-условий для данного помещения, где выполнялся дипломный проект, рекомендуем следующее:

- для голодного переуда года: температура воздуха от+20 до $+25^0$ С влажность воздуха от 45 % до 50 %; скорость движения воздуха 0,2 м/с;

- для жаркого переуда: температура воздуха от+20 до +27^0 С; влажность воздуха от 45 % до 50 %; скорость движения воздуха 0,5 м/с;
- допустимые нормы: температура воздуха от +18 до +27^{0}С; влажность воздуха от 45% до 70%; скорость движения воздуха от 0,3 м/с до 0,5 м/с.

4.5. Защита от электромагнитных полей

Электромагнитные волны возникают при ускоренном движении электрических зарядов и представляют собой взаимосвязанное распространение в пространстве изменяющихся электрического и магнитного полей. Совокупность этих полей, неразрывно связанных друг с другом, называется электромагнитным полем. Электромагнитное поле характеризуется напряженностью электрического поля Е (вольт на метр. В/м) и напряженностью магнитного поля Н (ампер на метр, А/м). Величины Е и Н – векторные, их колебания происходит во взаимоперпендикулярных плоскостях.

Диапазоны электромагнитных излучений в области радиочастот: 30кГц 300 гц.

При нагреве человеческого организма в электромагнитном поле происходит отвод избыточной теплоты до плотности потока энергии I=10 mВт/см^2 . Это величина называется тепловым порогом, начиная с которого система терморегуляции не справляется с отводом генерируемого тепла, происходит перегрев организма человека, что негативно сказывается на его здоровье.

Воздействие электромагнитных полей с интенсивностью, меньшей теплового порога, также небезопасно для здоровья человека. Оно нарушает функции сердечно – сосудистой системы, ухудшает обмен веществ приводит к изменению состава крови, снижает биохимическую активность белковых молекул. При длительном воздействии на работающих электромагнитного излучения различной частоты возникают повышенная утомляемость,

сонливость или нарушение сна, боли в области сердца, торможение рефлексов и т.д.

При воздействии на организм человека постоянных магнитных и электростатических полей с интенсивностью, превышающей безопасность уровень, могут развиться нарушения в деятельности сердечно-сосудистой системы, органов дыхания и пищеварения, возможно изменение состава крови и других. Электрические поля промышленной частоты (f=50 Гц) воздействуют на мозг и центральную нервную систему.

Предельно допустимые уровни облучения в диапазоне радиочастот определяются государственному стандарту «Электромагнитные поля радиочастот. Допустимые уровни на рабочих местах и требования к проведению контроля» в соответствии с этим нормативным документом установлена предельно допустимая напряженность электрического поля ($E_{пд}$, В/м) в диапазоне 0,06-300 МГц и предельно допустимая энергетическая нагрузка за рабочий день [$ЭН_{Eпд}$, $(В/м)^2$. ч].

Между этим величинами существует связь:

$$E_{ng} = \sqrt{\frac{ЭН_{Eng}}{T}}$$

где Т – время воздействия в течение рабочего дня, ч.

Для частот 0,06 ÷3,0 МГц: $E_{пд}$ =500 В/м, $ЭН_{Eпд}$ =20000 $(В/м)^2$.ч.

Для частот 3 ÷30 МГц: $E_{пд}$=300 В/м, $ЭН_{Eпд}$ =7000 $(В/м)^2$.ч.

Для частот 30 – 300 МГц: $E_{пд}$ =80 В/м, $ЭН_{Eпд}$=800 $(В/м)^2$.ч.

Предельно допустимая напряженность магнитного поля в диапазоне частот 0,06 – 3 МГц должна составлять $H_{пд}$=50 А/м. Взаимосвязь этой характеристики и предельно допустимой энергетикой нагрузки за рабочий день

$$H_{ng} = \sqrt{\frac{ЭН_{Hng}}{T}}$$

$[ЭН_{Нпд}, (А/м)^2.ч]$ имеет вид:

где Т – время воздействия, ч (величина $ЭН_{пд}$ не должна перевешать 200 $(А/м)^2.ч$).

Напряженность такого поля (Н) не должна превышать 8000 А/м.

Электрические поля промышленной частоты нормируются в соответствии с государственному стандарту «Электрическое поля промышленной частоты. Допустимые уровни напряжения и требования к проведению контроля на рабочих местах». В соответствии с этим нормативными документам предельно допустимый уровень напряженности электрического поля (Е) составляет 25000 В/м. Кроме того, оговаривается допустимое время пребывания (Т, ч) в электрическом поле с различной напряженностью:

$E \leqslant 5000$ В/м – в течении рабочего дня; $E = 5000 \div 20000$В/м, $T = \frac{50}{E} - 2;$ от 20000 до 25000 В/м, 1/6 часть рабочего дня.

В нашей стране разработаны также гигиенические нормативы для электростатических полей, электрических полей диапазона частот $1 \div 12$ кГц, магнитных полей промышленной частоты (50 Гц) и другие.

К основным методам защиты от электромагнитных излучений относятся: рациональное размещение излучающих и облучающих объектов, исключающее или ослабляющее воздействие излучения на персонал; ограничение места и времени нахождения работающих в электромагнитном поле; защита расстоянием; уменьшение мощности источника излучений; использование поглощающих и отражающих экранов; применение средств индивидуальной зашиты и некоторые другие.

Наиболее распространенный из них – экранирование рабочих мест, или непосредственно источника излучения. Отражающие экраны изготавливают из материалов с низким электросопротивлением (Си, латуни, Al и его сплавов, стали).

Экраны должны быть заземлены.

4.6. Расчет освещенности

Правильно спроектированное и выполненное производственное освещение улучшает условия зрительной работы, снижает утомляемость, способствует повышению производительности труда, благотворно влияет на производственную среду, оказывая положительное психологическое воздействие на работающего, повышает безопасность труда и снижает травматизм.

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Исходя из этого, рассчитаем параметры искусственного освещения.

Обычно искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые по сравнению с лампами накаливания имеют ряд существенных преимуществ:

- по спектральному составу света они близки к дневному, естественному;
- обладают более высоким КПД (в 1,5÷2 раза выше, чем КПД ламп накаливания);
- обладают повышенной светоотдачей (в 3÷4 раза выше, чем у ламп накаливания);
- более длительный срок службы.

Расчет освещения производится для комнаты площадью 15 м^2 ширина которой 5 м, высота –3м. Воспользуемся методом светового потока.

Для определения количества светильников определим световой поток,

$$F=\frac{E \cdot K \cdot S \cdot Z}{n}$$

подающий на поверхность по формуле:

где F –рассчитываемый световой поток, Лм;

E –нормирования минимальная освещенность, Лк. Работу программиста, в соответствии с таблицей, можно отнести к разделу точных работ, следовательно, минимальная освещенность будет E=300 Лк;

S –площадь освещаемого помещения (в нашем случае $S=15м^2$);

Z –отношение средней освещенности к минимальной (обычно принимается равным 1,1…1,2, пусть Z=1,1);

K –коэффицент запаса (в нашем случае К-1,1);

n –коэффицент использования.

Значение n определим по таблице коэффицентов использования различных светильников. Для этого вычислим индекс помещения по формуле:

$$I = \frac{S}{h\ (A+B)}$$

где

S –площадь помещения, $S=15\ м^2$; h –расчетная высота подвеса, h=2,92м; A –ширина помещения, A=3м, B –длина, B=5м.

$$I = \frac{15}{2,92\ (3+5)} = 0,64$$

Подставив значения получим:

Зная индекс помещения I, по таблице находим n=0,22

Подставим все значения в формулу для определения светового потока F:

$$F = \frac{300\ \ 1,5\ \ 15\ \ 1,1}{0,22} = 33750\ \ Лм$$

Для освещения выбираем люминесцентные лампы типа ЛБ40-1, световой поток которых F=4320 Лк.

Рассчитаем необходимое количество ламп по формуле:

$$N=\frac{F}{F_л}$$

где N –определяемое число ламп;

F –световой поток, F=33750 Лм;

$F_л$ -световой поток лампы, $F_л$ =4320 Лм.

$$N=\frac{33750}{4320}=8 \text{ шт}$$

При выборе осветительных приборов используем светильники типа ОД. Каждый светильник комплектуется двумя лампами.

4.7. Пожаробезопасность

Пожар – неконтролируемое горение вне специального очага, наносящее материальный ущерб. Возможности создания условий для возникновения пожара или его быстрого развития представляют собой пожарную опасность.

Опасными поражающими факторами пожара являются:

- открытый огонь и искры;
- выделяющееся при горении тепло вызывает повышение температуры окружающей среды, и когда она доходит до критической для окружающих предметов и вещей, загораются и они. Очаг пожара разрастается;
- токсичные продукты горения, дым;
- падающие части строительных конструкций и агрегатов.

Основными факторами взрыва являются:

– воздушная взрывная волна, основным параметром которой является избыточное давление в ее фронте;
– осколочные поля, создаваемые летящими обломками взрывающихся объектов, порождающее действие которых определяется количеством летящих осколков, их кинетической энергией и радиусом разлета.

Согласно санитарному норму производственное здание относится к категории Г (производства, связанные с обработкой несгораемых веществ и материалов в горячем, раскаленном или расплавленном состояние, сопровождающиеся выделением лучистого тепла, искр и пламени). Использование в компьютерах мощных радиоэлементов: понижающие и высоковольтные трансформаторы, выпрямляющие диоды, транзисторы, микросхемы, токопроводящие линии, работающих при относительно больших токах и напряжениях, предполагает их нагрев до высоких температур. Тепловое действие электрического тока, проходящего по проводам, при неисправностях или перегрузка электроустановка или аппаратуры может быть причиной пожара.

Основными причинами пожара являются:
1) повреждение изоляции проводов; 2) попадание на неизолированные провода токопроводящих предметов; 3) воздействие на провода химически – активных веществ, паров; 4) неправильный монтаж установки прибора.

Зашита электрооборудования осуществляется при помощи плавких предохранителей и специальных автоматов, включаемых последовательно в электрическую цепь. При конструировании аппаратуры, где возможно образование искры электродуги, предусмотрены кожухи или дугогасящие решетки.

На производстве применяется метод и средства пожаротушения; применения углекислотных огнетушителей, так как CO_2 не портит оборудование и не проводит электрический ток.

В случае пожара имеются эвакуационные путы: 1) из помещений первого этажа – наружу непосредственно или через коридор, вестибюль, лестничную

клетку; 2) из помещений любого этажа – в коридор или проход ведущий к лестничной клетки, или на лестничную клетку, имеющую выход наружу, отдельной от примыкающих коридоров перегородками с дверьми;

3) из помещения – в соседние помещение на том же этаже, обеспеченные выходом непосредственно наружу, через коридор, лестничную клетку или вестибюль.

В производственном здании имеются планы эвакуации, в которых указаны пути эвакуации людей из здания в случаях возгорания.

Заключение

В Web-дизайне нет жестких правил. Поскольку главная наша задача – сделать содержимое страницы доступным для максимального количества пользователей, то для продвижения вперед одинаково важны и эксперимент, и использование новых технологий с учетом существующих реалий.

В обзорной части выпускной квалификационном работы был проанализирован язык разметки гипертекстовых документов HTML, его основные функции свойства и параметры. Такие технологии, как JavaScript, были затронуты лишь поверхностно дабы показать эффективность совокупности использования HTML с интерактивными скриптовыми технологиями.

В выпускной квалификационной работе разработаны примеры программ для проведение практических занятий по предмету "Web программирования" для студентов направления 5521900-Информатика и информационные технологии.

Литература

1. Будилов В. А. JavaScript, XML и объективная модель документа : монография. -СПб: Наука и Техника, 2001.-352 с.
2. Николенко Д. В. Практические занятия по JavaScript : для начинающих. - СПб: Наука и техника, 2000.-352 с.
3. Дарнелл Рик. JavaScript: Справочник. -СПб; М.;Харьков: Питер, 2001.-192 с.
4. Леонтьев Б. Web- дизайн : Руководство для пользователя/ Б. Леонтьев. - М.: Познават. кн. плюс, 2001.-320 с.
5. Румянцев Д. Сам себе WEB программист: Практикум создания качеств. WEB-сайта/ Д. Румянцев. -М.: ИНФРА-М, 2001.-207 с.
6. Айзекс А. Dynamic HTML BHV-Санкт-Петербург, 1998
7. Ганчаров А. Самоучитель HTML. Питер, 2000
8. Дарнелл Р. HTML 4 Энциклопедия пользователя. ДиаСофт 1999
9. Денисов Internet Explorer 5. Справочник. Питер, 1999
10. Хоумер А. Dynamic HTML. Справочник. Питер, 1999
11. Петюшкин А.В., HTML. Экспресс-курс. – СПб.: БХВ - Петербург, 2003
12. Кингсли-Хью Э., JavaScript: учебный курс. – СПб.: Питер, 2002
13. www.intuit.ru - Интернет-университет информационных технологий.
14. http://www.robotland.ru/

Приложение

Листинг сайта:

Index.htm

```
<HTML><HEAD><TITLE>Примеры задач для проведение практических занятий по предмету

&quot;Web программирования&quot;</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1251">
<META content=default name=Author>
<META content="MSHTML 6.00.2900.2876" name=GENERATOR>
<style type="text/css">
<!--
.style1 {
	font-size: medium;
	font-weight: bold;
}
.style2 {color: #FFFFFF}
-->
</style>
</HEAD>
<BODY text=#0000ff vLink=#3366ff aLink=#ff0000 link=#3366ff bgColor=#ffffff>
<CENTER>
<table width="100%" border="0">
  <tr>
    <td bgcolor="#48A6D9"><div align="center" class="style1">
      <p class="style2">Примеры задач для проведение практических занятий </p>
```

```
            <p class="style2">по предмету &quot;Web программирования&quot;
для студентов </p>
            <p class="style2">направления 5521900-Информатика и
информационные технологии</p>
          </div></td>
        </tr>
        <tr>
          <td><hr></td>
        </tr>
      </table>
      <H2>Перечень заданий </H2>
      </CENTER>
      <p><A
        href="pr1.htm">Задание №1. Проверка правильности электронного
адреса (по заданному

      шаблону)</A> </p>
      <p><A
        href="pr2.htm">Задание №2. Проверка заполнения полей формы
(проверка на наличие

      данных)</A> </p>
      <p><A
        href="pr3.htm">Задание №3. Пример формы заказа (объединение всех
проверок на уровне

      формы)</A> </p>
      <p><A
        href="pr4.htm">Задание №4. Фоновая музыка</A> </p>
      <p><A
```

```
href="pr6.htm">Задание №5. Простая анимация на JavaScript</A> </p>
<p><A
href="pr7.htm">Задание №6-а. Пример таблицы стилей(без
использования JavaScript)</A>

</p>
<p><A
href="pr8.htm">Задание №6-б. Применение динамических стилей,
используя JavaScript</A>

</p>
<p><A
href="pr9.htm">Задание №7. Управление слоями, используя
JavaScript</A> </p>
<p><A
href="pr10.htm">Задание №8. Создание анимации с помощью
DHTML</A> </p>
<p><A
href="begin.htm">Задание №9. ПОИСКОВАЯ СИСТЕМА (написана на
JAVASCRIPT)</A></p>
<p><A
href="primer.htm">Дополнительные примеры</A></p>
<BR>  <BR>  <!--
<p><b><font color="#666666"><font size=-1><a href="we.html">o

дизайнерах</a>       
<a href="curs.html">от дизайнеров</a></font></font></b>
<br><b><font size=-1>Copyright</font><font face="Symbol">
У</font><font size=-1>
1999  Бердников И. Ю. &amp; Мосунов Д. Ю.</font></b></center>
```

```
 --></BODY></HTML>
```

pr1.htm

```
<HTML><HEAD><TITLE>Пример "промывки" данных и установления обратной связи</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1251">
<SCRIPT language=JavaScript>

////////////////////////////////////////////////////
// Поскольку все верные e-mail адреса содержат
// точку и символ @, эта функция ищет эти
// два символа.  Если они обнаружены, e-mail
// адрес считается верным; если нет, пользователя
// приглашают ввести адрес e-mail повторно.
////////////////////////////////////////////////////

function isAValidEmail(inputValue) {

   var foundAt = false
   var foundDot = false
   var atPosition = -1
   var dotPosition = -1

   // Проверить один за другим все символы адреса
   // e-mail и выставить флаг когда (и если)
   // знак @ и точка обнаружены.

   for (var i=0; i<=inputValue.length; i++) {
```

```
            if (inputValue.charAt(i) == "@" ) {
               foundAt = true
               atPosition = i
             }
            else if (inputValue.charAt(i) == ".") {
               foundDot = true
               dotPosition = i
             }
           }

           // Если оба символа - @ и точка - обнаружены и
           //  нужном порядке (символ @ должен идти первым)...
           if ((foundAt && foundDot) && (atPosition < dotPosition)) {
              // Указан верный формат адреса e-mail
              alert("Спасибо за верный ввод адреса e-mail!")
              return true
           }
           else {
              // Адрес e-mail неверен
               alert("Извините, вы неверно ввели адрес e-mail. Пожалуйста,
попробуйте еще раз.")
              return false
           }
        }


        </SCRIPT>

        <META content="MSHTML 6.00.2900.2876"
name=GENERATOR></HEAD>
```

```
 <BODY>
<H1>Пример "промывки" данных и установления обратной связи</H1>
<OL>
 <LI>Введите заведомо неверный адрес e-mail (например, такой как
<I>XYZ</I>) в
 поле ввода внизу страницы. Когда закончите, щелкните <B>здесь</B>.
 <P></P>
 <LI>Затем, введите верный адрес e-mail (такой как
<I>emily@emilyv.com</I>) в
 поле и щелкните <B>здесь</B>. </LI></OL>
<P>
<FORM name=myForm>Адрес E-mail: <BR><INPUT
onblur=isAValidEmail(this.value)
size=25 name=emailAddress> </FORM></P></BODY></HTML>
```

pr2.htm

```
<HTML><HEAD><TITLE>Пример формы заказа</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-
1251">
<SCRIPT language=JavaScript>

//////////////////////////////////////////
// Returns true if a value contains any
// characters; otherwise, returns false.
//////////////////////////////////////////
function exists(inputValue) {

   var aCharExists = false

   // Step through the inputValue, using the charAt()
```

```
        // method to detect non-space characters.

       for (var i=0; i<=inputValue.length; i++) {
        if (inputValue.charAt(i) != " " && inputValue.charAt(i) != "") {
           aCharExists = true
           break
        }
       }

       return aCharExists
    }

    </SCRIPT>

    <META content="MSHTML 6.00.2900.2876"
name=GENERATOR></HEAD>
    <BODY>
    <H1>Пример формы заказа</H1>Щелкните на поле для ввода, а затем - в любом другом
    месте (не вводя никаких данных), чтобы проверить, работает ли функция exists().
    <P>
    <FORM name=myForm>Имя: <INPUT
    onblur="if (!exists(this.value)) { alert('Пожалуйста, введите ваше имя'); }"
    size=25 name=firstName>
    <P>Фамилия: <INPUT
    onblur="if (!exists(this.value)) { alert('Пожалуйста, введите фамилию') }"
    size=35 name=lastName> </FORM></P></BODY></HTML>
```

pr3.htm

```
 <HTML><HEAD><TITLE>Пример формы заявки</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1251">
<SCRIPT language=JavaScript>
<!-- скрыть этот сценарий от броузеров, не поддерживающих JavaScript

//////////////////////////////////////////////////////
// Проверка на наличие нечислового значения.
//////////////////////////////////////////////////////
function isANumber(inputValue){

   // Предположим, все в порядке
   var result = true

     // Если функция parseFloat() возвращает значение
     // false, значит на первой позиции находится не
     // численное значение
   if (!parseFloat(inputValue)) {
      result = false
   }

        // В противном случае, все равно следует проверить
        // оставшиеся цифры в inputValue, символ за символом,
        // и установить значение result = false, если будет
    // найдено любое не численное значение.
   else {
      for (var i=0; i<inputValue.length; i++) {
        if (inputValue.charAt(i) != " ") {
           if (!parseFloat(inputValue.charAt(i))) {
              result = false
              break
```

```
            }
        }
    }
}

    // Вернуть значение true (inputValue это правильный
    // номер) или false (неправильный).
  return result
}

////////////////////////////////////////////////////
// Проверить, содержит ли введенное значение
// символы "@" и "."
////////////////////////////////////////////////////
function isAValidEmail(inputValue) {

  var foundAt = false
  var foundDot = false

  // Просмотреть значение inputValue в поисках
  // символов "@" and "."

  for (var i=0; i<=inputValue.length; i++) {
    if (inputValue.charAt(i) == "@" ) {
      foundAt = true
    }
    else if (inputValue.charAt(i) == ".") {
      foundDot = true
    }
  }
```

```
    // Если найдены оба символа: @ и точка, полагаем,
    // что адрес e-mail введен правильно
    // в противном случае возвращается
    // значение false, чтобы было понятно,
    // что адрес e-mail введен неправильно.

    if (foundAt && foundDot) {
        return true
    }
    else {
        return false
    }
}

//////////////////////////////////////////////////////
// Проверить, чтобы введенное значение содержало
// 10 или более цифр.  Используя такой подход,
// пользователи будут вводить номер телефона по
// принятому в США шаблону, например, (123)456-7890,
// или в европейском стиле: 123.456.7890
//////////////////////////////////////////////////////
function isAValidPhoneNumber(inputValue) {
    var digitsFound = 0

    // Просмотр значения inputValue для определения
    // количества содержащихся в нем цифр

    for (var i=0; i<=inputValue.length; i++) {
      if (isANumber(inputValue.charAt(i))) {
```

```
            digitsFound++
        }
    }

    // Если значение inputValue содержит по крайней мере
    // 10 цифр, полагаем, что номер введен правильно
    if (digitsFound >= 10) {
        return true
    }
    else {
        return false
    }
}

//////////////////////////////////////////////////////
// Проверка на наличие символов. (Пробелы
// не считаются.)
//////////////////////////////////////////////////////

function exists(inputValue) {

    var aCharExists = false

    // Проверка значения inputValue с помощью
    // метода charAt() для поиска символов
    // (отличных от пробелов).

    for (var i=0; i<=inputValue.length; i++) {
      if (inputValue.charAt(i) != " " && inputValue.charAt(i) != "") {
          aCharExists = true
```

```
            break
        }
    }

    return aCharExists
}

////////////////////////////////////////////////////
// Выполняет проверку полей, которая не может быть
// выполнена до того, как будут введены все данные.
////////////////////////////////////////////////////
function validateForm() {

    var rc = true
        // Проверка №1 зависимого поля: проверка выбора категории услуг
    ////////////////////////////////////////////////////
    // Посетителям следует выбрать один из вариантов,
    // чтобы правильно заполнить заявку, в зависимости от
    // того, в чем они заинтересованы: услуги по дизайну,
    // поддержке или продвижению.
////////////////////////////////////////////////////

    if (!document.quoteForm.designChoice.checked &&
        !document.quoteForm.maintChoice.checked &&
        !document.quoteForm.promoChoice.checked) {
            alert("Пожалуйста, выберите интересующие вас услуги: дизайн,
поддержка или продвижение, чтобы мы могли точно выполнить ваш заказ.
Спасибо!")
        rc = false
    }
```

```
        // Проверка №2 зависимого поля: проверка
        // заполнения названия организации, если пользователь
        // выбрал параметр "служащий"
   ////////////////////////////////////////////////////
   // Если посетители являются служащими, следует
   // указать название своей организации.
////////////////////////////////////////////////////

   if (document.quoteForm.bizChoice[1].checked) {
      if (!document.quoteForm.corpName.value) {
         alert("Вы указали, что вы — служащий; не могли бы вы указать название
вашей организации?  Спасибо!")
         rc = false

      }
   }
        // Проверка №3 зависимого поля: двойная проверка
        // полей для заполнения имени и фамилии

   ////////////////////////////////////////////////////
// Посетителям следует указать свое имя и фамилию.
////////////////////////////////////////////////////

   if (!document.quoteForm.firstName.value ||
      !document.quoteForm.lastName.value) {
      alert("Пожалуйста, укажите ваше полное имя и фамилию.  Спасибо!")
      rc = false
   }
```

```
        // Проверка №4 зависимого поля: проверка
        // заполнения электронного адреса или номера телефона
   ////////////////////////////////////////////////////
   // Посетителям следует указать адрес e-mail или
   // номер телефона.
////////////////////////////////////////////////////
   if (!document.quoteForm.emailChoice.checked &&
      !document.quoteForm.phoneChoice.checked) {
      alert("Пожалуйста, укажите, как с вами связаться: по электронной почте
или телефону.  Спасибо!")
      rc = false
   }
       // Проверка №5 зависимого поля: проверка наличия
       // адреса e-mail (если пользователь
       // выбрал параметр "электронная почта")

   ////////////////////////////////////////////////////
   // Если посетитель указывает, что он предпочитает
   // общение по электронной почте, вывести сообщение,
   // если он забудет ввести адрес e-mail
   // (то же самое при выборе параметра "телефон")
////////////////////////////////////////////////////
   if (document.quoteForm.emailChoice.checked &&
      !isAValidEmail(document.quoteForm.emailAddr.value)) {
         alert("Мы не сможем связаться с вами по электронной почте, пока вы не
введете свой адрес.  Спасибо!")
      rc = false
   }
   else {
      if (document.quoteForm.phoneChoice.checked &&
```

```
       !isAValidPhoneNumber(document.quoteForm.phoneNumber.value)) {
         alert("Мы не сможем связаться с вами по телефону, пока вы не введете
свой номер (убедитесь, что вы указали свой код).  Спасибо!")
      rc = false
      }
  }

  if (rc) {
     // Если переменная rc содержит не нулевое
     // значение, тогда данные будут переданы!
     alert("Спасибо! Мы свяжемся в вами в ближайшее время.")
  }
  return rc
}

// -->
</SCRIPT>

<META content="MSHTML 6.00.2900.2876" name=GENERATOR></HEAD>
<BODY>
<H1>Пример формы заявки</H1>
<HR>

<TABLE cellSpacing=0 cellPadding=10 width="100%" border=0>
 <TBODY>
 <TR>
   <TD vAlign=top><FONT face="Arial, Helvetica, Verdana" size=5>
     <CENTER>Заполните заявку на разработку Web дизайна или продвижения
вашего
```

```
Web-узла сайта от компании
<B>Webmeister</B></CENTER><BR></FONT>
<P><FONT face="Arial, Helvetica, Verdana" size=2>Каждый проект
отличается
своей индивидуальностью, поэтому мы можем прийти к окончательному
соглашению, только если предварительно побеседуем с вами и просмотрим
материалы.
<P>Будем благодарны, если вы ответите на некоторые вопросы. Заполните
необходимую информацию и щелкните на кнопке "Отправить заявку", и мы
свяжемся с вами по электронной почте (или если вы предпочитаете, по
телефону) в течении одного рабочего дня.
</P></FONT></TD></TR></TBODY></TABLE>
<P>
<HR>

<TABLE cellSpacing=10 cellPadding=10 width="100%" border=0>
 <TBODY>
 <TR>
  <TD>
   <FORM name=quoteForm onsubmit="return validateForm();">
   <P><B><FONT face=Arial size=2>В каком виде услуг вы заинтересованы?
   </FONT></B><FONT face=Arial size=2>(Выберите все, что вам подходит.)
   </FONT>
   <P><FONT face=Arial size=2>Дизайн Web-сайта <INPUT type=checkbox
   value=design name=designChoice>      
Поддержка
   Web-сайта <INPUT type=checkbox value=maint
   name=maintChoice>      Реклама Web-сайта
<INPUT
   type=checkbox value=promo name=promoChoice> </FONT>
```

<P><B><FONT face=Arial size=2>Перечислите причины создания Web-сайта.

(Если у вас уже имеется Web-сайт, укажите, в каких целях вы его

используете?) </FONT></B><FONT face="Helvetica, Arial, Verdana" size=2>

<P><TEXTAREA name=purpose rows=5 wrap=VIRTUAL cols=60></TEXTAREA> </FONT>

<P><B><FONT face=Arial size=2>Вы хотите разместить на своем Web-узле

фотографии? </FONT></B><FONT face=Arial size=2>
yes <INPUT type=radio

CHECKED value=hasPix name=pixChoice> no <INPUT type=radio value=hasNoPix

name=pixChoice> </FONT>

<P><B><FONT face=Arial size=2>Вы хотите рекламировать/продавать на своем

Web-узле какую-нибудь продукцию? </FONT></B><FONT face=Arial size=2>
Да

<INPUT type=radio CHECKED value=hasProducts name=cdChoice> Нет <INPUT

type=radio value=hasNoProducts name=cdChoice> </FONT>

<P><B><FONT face=Arial size=2>У вас собственный бизнес или вы являетесь

служащим организации? </FONT></B><FONT face=Arial size=2>
Имею свой

бизнес <INPUT type=radio CHECKED value=isOwner name=bizChoice> Служащий

<INPUT type=radio value=isEmployee name=bizChoice> </FONT>

<P><B><FONT face=Arial size=2>Если вы работаете в организации, не могли бы

вы указать ее название? </FONT></B><FONT face=Arial size=2><INPUT size=25

```
 name=corpName> </FONT>
<P><B><FONT face=Arial size=2>Располагаете ли вы дополнительной
информацией, которую нам следует учесть при оформлении вашей заявки?
</FONT></B><FONT face="Helvetica, Arial, Verdana" size=2>
<P><TEXTAREA name=extraInfo rows=5 wrap=VIRTUAL
cols=60></TEXTAREA>
<P></FONT>
<TABLE>
 <TBODY>
 <TR>
  <TD><B><FONT face=Arial size=2>Ваше имя: </FONT></B></TD>
  <TD><B><FONT face=Arial size=2>Ваша фамилия:
</FONT></B></TD></TR>
 <TR>
  <TD><FONT face=Arial size=2><INPUT
   onblur="if (!exists(this.value)) { alert('Пожалуйста, введите свое имя'); }"
   size=25 name=firstName> </FONT></TD>
  <TD><FONT face=Arial size=2><INPUT
   onblur="if (!exists(this.value)) { alert('Пожалуйста, введите свою
фамилию') }"
   size=35 name=lastName> </FONT></TD></TR>
 <TR>
  <TD colSpan=2><B><FONT face=Arial size=2>Какой способ связи вы
   предпочитаете (электронная почта, телефон, или оба)?
 </FONT></B></TD></TR>
 <TR>
  <TD align=right><FONT face=Arial size=2>электронная почта<INPUT
   type=checkbox value=email name=emailChoice> </FONT></TD>
  <TD><FONT face=Arial size=2><INPUT size=35 name=emailAddr>
 </FONT></TD></TR>
```

<TR>
<TD align=right><FONT face=Arial size=2>телефон <INPUT type=checkbox
value=phone name=phoneChoice> </FONT></TD>
<TD><FONT face=Arial size=2><INPUT size=15 name=phoneNumber>
</FONT></TD></TR></TBODY></TABLE><FONT face="Helvetica, Arial, Verdana"
size=2>
<P>
<CENTER><INPUT type=submit value="Отправить заявку">
      <INPUT type=reset value="Очистить форму">
</CENTER></FORM>
<HR>
</FONT></TD></TR></TBODY></TABLE></BODY></HTML>

pr4.htm

```
<HTML><HEAD><TITLE>Фоновая музыка</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1251">
<META content="MSHTML 6.00.2900.2876" name=GENERATOR></HEAD>
<BODY>
<H1>Фоновая музыка</H1>
<P>Эта страница имеет фоновую музыку(для браузеров Explorer 3.0 и выше или
Netscape ).</P>
<SCRIPT language=JavaScript>
if (navigator.appName.indexOf("Microsoft") > -1)  document.write("<BGSOUND
SRC='10.mp3'>");
if (navigator.appVersionName.indexOf("Netscape4") > -1)
document.write("<EMBED SRC='10.mp3'>");
```

```
 </SCRIPT>
</BODY></HTML>
```

pr6.htm

```
<HTML><HEAD><TITLE>Простая анимация на JavaScript</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1251">
<SCRIPT language=JavaScript>
var cbox=0;
var nbox=1;
var cimage=0;
var nimage=0;

function preload() {
a1 = new Image();
a1.src = "mouse1.gif";
a2 = new Image();
a2.src = "mouse2.gif";
a3 = new Image();
a3.src = "mouse3.gif";
a4 = new Image();
a4.src = "mouse4.gif";
a5 = new Image();
a5.src = "mouse5.gif";
a6 = new Image();
a6.src = "mouse6.gif";
a7 = new Image();
a7.src = "mouse7.gif";
a8 = new Image();
a8.src = "mouse8.gif";
window.setTimeout("next();",500);
```

```
 }

function next() {
cimage += 1;
if (cimage > 8) {
        cimage = 4;
        document.images[cbox].src = "mouse0.gif";
                cbox = (cbox + 1) % 5;
                nbox = (cbox + 1) % 5;
    }
nimage = cimage - 5;
if (nimage <= 0) nimage = 0;
document.images[cbox].src = "mouse" + cimage + ".gif";
document.images[nbox].src = "mouse" + nimage + ".gif";
window.setTimeout("next();",100);
}
</SCRIPT>

<META content="MSHTML 6.00.2900.2876" name=GENERATOR></HEAD>
<BODY onload=preload();>
<H1>Анимация на JavaScript</H1>
<HR>

<CENTER><IMG height=100 alt="" src="pr6.files/mouse0.gif" width=100
border=0>
<IMG height=100 alt="" src="pr6.files/mouse0.gif" width=100 border=0> <IMG
height=100 alt="" src="pr6.files/mouse0.gif" width=100 border=0> <IMG
height=100
alt="" src="pr6.files/mouse0.gif" width=100 border=0> <IMG height=100 alt=""
src="pr6.files/mouse0.gif" width=100 border=0> </CENTER>
```

```
 <HR>
</BODY></HTML>
```

pr7.htm

```
<HTML><HEAD><TITLE>Пример таблицы стилей</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1251">
<STYLE>BODY {
        COLOR: blue
}
P {
        MARGIN-LEFT: 20%; MARGIN-RIGHT: 20%; TEXT-ALIGN: center
}
H1 {
        COLOR: red
}
H2 {
        COLOR: red
}
H3 {
        COLOR: red
}
UL {
        FONT-WEIGHT: bold; COLOR: green
}
</STYLE>

<META content="MSHTML 6.00.2900.2876" name=GENERATOR></HEAD>
<BODY>
<H1>Добро пожаловать на нашу страницу</H1>
<P>Заголовок имеет красный цвет, т.к. мы определили заголовки H1-H3 цветом
```

"red". Абзац - синий (blue) см.body. Абзац выровнен по центру и имеет левое и правое поле 20% от его размера. </P>
<P style="COLOR: black">Этот абзац содерхит черный текст. Основной стиль был
заменен стилем элемента. Изменен только цвет текста, выравнивание остается то
же. </P>
<UL>
<LI>Это маркированный список.
<LI>Текст полужирный, зеленого цвета. Это парметры определнные по умолчанию
для таких списков.
<LI style="COLOR: red">Этот элемент имеет красный маркер. Стиль по умолчанию
для него изменен.
<LI>Этот пункт со стилем по умолчанию (цвет - зеленый). </LI></UL>
<P>Этот абзац со стилем по умолчанию (цвет - синий).</P></BODY></HTML>

pr8.htm

<HTML><HEAD><TITLE>Управление стилями</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1251">
<META content="MSHTML 6.00.2900.2876"
name=GENERATOR></HEAD><FRAMESET
rows=*,*><FRAME name=topframe src="pr8.files/list17-4r.htm"><FRAME
name=botframe
src="pr8.files/list17-5r.htm"></FRAMESET></HTML>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

```
 <!-- saved from
url=(0077)http://www.math.omsu.omskreg.ru/info/learn/js/speckurs/primer16/list17-
4r.htm -->
<HTML><HEAD><TITLE>Изменение стилей JavaScript</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1251">
<META content="MSHTML 6.00.2900.2876" name=GENERATOR></HEAD>
<BODY>
<H1>Управление стилями JavaScript</H1>
<HR>
Укажите цвет текста абзаца и заголовка и щелкните на кнопке "Reload"
(обновить).
Указанные параметры будут применены к нижнему документу.
<FORM name=form1><B>Цвет тексат абзаца: </B><SELECT name=body>
<OPTION
 value=red selected>Red</OPTION> <OPTION value=blue>Blue</OPTION>
<OPTION
 value=green>Green</OPTION> <OPTION value=yellow>Yellow</OPTION>
<OPTION
 value=black>Black</OPTION></SELECT> <BR><B>Цвет текста заголовка:
</B><SELECT
name=heading> <OPTION value=red selected>Red</OPTION> <OPTION
 value=blue>Blue</OPTION> <OPTION value=green>Green</OPTION>
<OPTION
 value=yellow>Yellow</OPTION> <OPTION
value=black>Black</OPTION></SELECT> <INPUT
onclick=parent.botframe.location.reload(); type=button value=Reload>
</FORM></BODY></HTML>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

```
 <!-- saved from
url=(0077)http://www.math.omsu.omskreg.ru/info/learn/js/speckurs/primer16/list17-
5r.htm -->
<HTML><HEAD><TITLE>Применение динамических стилей, используя
JavaScript</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1251">
<STYLE id=iestyle type=text/css></STYLE>

<SCRIPT language=JavaScript>
  if ((navigator.appVersion.charAt(0) == "4")
  ||(navigator.appVersion.charAt(0) == "5")) {
     if (navigator.appName == "Netscape")
        browser="netscape";
     else if (navigator.appName=="Microsoft Internet Explorer")
        browser="ie";
     else alert("This document requires a 4.0 or later browser.");
  }
  i = parent.topframe.document.form1.heading.selectedIndex;
  hc = parent.topframe.document.form1.heading.options[i].value;
  i = parent.topframe.document.form1.body.selectedIndex;
  dc = parent.topframe.document.form1.body.options[i].value;
  if (browser=="netscape") {
     document.tags.BODY.color = dc;
     document.tags.H1.color = hc;
  }
  if (browser=="ie") {
     hc1 = "color:" + hc;
     dc1 = "color:" + dc;
     document.styleSheets["iestyle"].addRule("H1",hc1);
     document.styleSheets["iestyle"].addRule("BODY",dc1);
```

```
   }
</SCRIPT>

<META content="MSHTML 6.00.2900.2876" name=GENERATOR></HEAD>
<BODY>
<H1>Это заголовок.
<SCRIPT language=JavaScript>
document.write(" " + hc + ".")
</SCRIPT>
 </H1>
<P>Это основной текст абзаца.</P>
<P>Это второй абзац.
<SCRIPT language=JavaScript>
document.write(" Это основной текст " + dc + ".");
</SCRIPT>
 </P></BODY></HTML>
```

pr9.htm

```
<HTML><HEAD><TITLE>Управление слоями</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1251">
<SCRIPT language=JavaScript>
 function showOrHide(value) {
    if (value==0) {
        if (document.layers)
           document.layers["layer3"].visibility='hide';
        else
           document.all["layer3"].style.visibility='hidden';
   }
   else if (value==1) {
      if (document.layers)
```

```
        document.layers["layer3"].visibility='show';
      else
        document.all["layer3"].style.visibility='visible';
  }
}
</SCRIPT>

<META content="MSHTML 6.00.2900.2876" name=GENERATOR></HEAD>
<BODY>
<H1>Управление слоями, используя JavaScript</H1>
<FORM name=form1><INPUT onclick=showOrHide(1); type=button
value="Показать Layer"> <INPUT onclick=showOrHide(0); type=button
value="Скрыть Layer"> </FORM>
<P>Этот абзац обычного текста вне слоев.</P>
<DIV id=layer1 style="POSITION: static; BACKGROUND-COLOR:
lightgreen">Это
статический слой, расположенный поверх основного документа HTML. </DIV>
<P>Этот еще один абзац обычного текста.</P>
<DIV id=layer2
style="LEFT: 400px; WIDTH: 100px; POSITION: absolute; TOP: 200px; HEIGHT:
200px; BACKGROUND-COLOR: yellow">Это
первый слой. </DIV>
<DIV id=layer3
style="LEFT: 35px; VISIBILITY: hidden; WIDTH: 50px; POSITION: absolute;
TOP: 210px; HEIGHT: 50px; BACKGROUND-COLOR: orange">Этот
слой по умолчанию не отображается. Он отображается с помощью JavaScript.
</DIV>
<DIV id=layer4
style="LEFT: 100px; WIDTH: 200px; POSITION: relative; TOP: 100px; HEIGHT:
100px; BACKGROUND-COLOR: lightblue">Это
```

последний слой. Заметим, что слои отображаются так, как мы их определили.

```
</DIV></BODY></HTML>
```

pr10.htm

```
<HTML><HEAD><TITLE>Создание анимации с помощью DHTML</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1251">
<SCRIPT language=JavaScript>
 var pos1=-95;
 var pos2=-95;
 var pos3=-95;
 var speed1 = Math.floor(Math.random()*10)+2;
 var speed2 = Math.floor(Math.random()*10)+2;
 var speed3 = Math.floor(Math.random()*10)+2;
 function next() {
   pos1 += speed1;
   pos2 += speed2;
   pos3 += speed3;
   if (pos1 > 795) pos1 = -95;
   if (pos2 > 795) pos2 = -95;
   if (pos3 > 795) pos3 = -95;
   if (document.layers) {
      document.layers[0].left = pos1;
      document.layers[1].left = pos2;
      document.layers[2].left = pos3;
   }
   else {
      document.all.mouse1.style.left = pos1;
      document.all.mouse2.style.left = pos2;
      document.all.mouse3.style.left = pos3;
   }
```

```
    window.setTimeout("next();",10);
 }
 </SCRIPT>

<META content="MSHTML 6.00.2900.2876" name=GENERATOR></HEAD>
<BODY onload=next();>
<H1>Создание анимации с помощью DHTML</H1>
<HR>

<DIV id=mouse1
style="LEFT: 0px; WIDTH: 100px; POSITION: absolute; TOP: 100px; HEIGHT:
100px"><IMG
height=100 alt="" src="pr10.files/mouse5.gif" width=100 border=0> </DIV>
<DIV id=mouse2
style="LEFT: 0px; WIDTH: 100px; POSITION: absolute; TOP: 200px; HEIGHT:
100px"><IMG
height=100 alt="" src="pr10.files/mouse5.gif" width=100 border=0> </DIV>
<DIV id=mouse3
style="LEFT: 0px; WIDTH: 100px; POSITION: absolute; TOP: 300px; HEIGHT:
100px"><IMG
height=100 alt="" src="pr10.files/mouse5.gif" width=100 border=0>
</DIV></BODY></HTML>
```