МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ХАРЬКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ»

Н. В. БОРИСОВА
О. В. КАНИЩЕВА

ОСНОВЫ ВЕБ-ТЕХНОЛОГИЙ

Учебное пособие
для студентов специальности
«Прикладная лингвистика»

Харьков
НТУ «ХПИ»
2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ХАРЬКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ»

Н. В. БОРИСОВА
О. В. КАНИЩЕВА

ОСНОВЫ ВЕБ-ТЕХНОЛОГИЙ

Учебное пособие
для студентов специальности
«Прикладная лингвистика»

Утверждено
редакционно-издательским
советом университета,
протокол № 1 от 07.06.2013 года

Харьков
НТУ «ХПИ»
2016

УДК 004.738.5
ББК 32.973.202я73
Б 82

Рецензенты:
И. П. Шостак, д-р техн. наук, проф., проф. Национального аэрокосмического университета им. Н. Е. Жуковского «Харьковский авиационный институт»;
И. Д. Вечирская, канд. техн. наук, ст. науч. сотрудник, Харьковский национальный университет радиоэлектроники

Містить теоретичний матеріал, що охоплює широке коло питань, пов'язаних з веб-технологіями та веб-дізайном.

Призначено для студентів різних спеціальностей.

Борисова Н. В.

Б 82 Основы веб-технологий : учеб. пособ. / Н. В. Борисова, О. В. Канищева. – Х. : НТУ «ХПИ», 2016. – 229 с. – На рус. яз.

ISBN

Содержит теоретический материал, охватывающий широкий круг вопросов, связанных с веб-технологиями и веб-дизайном.

Предназначено для студентов различных специальностей.

Ил.: 40. Табл.: 7. Библиогр.: 13 назв.

УДК 004.738.5
ББК 32.973.202я73

ISBN

© Н. В. Борисова, 2016
© О. В. Канищева, 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Материал учебного пособия, представленный в настоящем издании, охватывает широкий круг вопросов, связанных с веб-технологиями.

Первый раздел посвящен языку разметки HTML, который позволяет задать местоположение элементов веб-страницы в окне веб-браузера. С помощью HTML можно отформатировать отдельные символы или целые фрагменты текста, вставить изображение, таблицу или форму, создать панель навигации с помощью карт-изображений, разделить окно веб-браузера на несколько областей, вставить гиперссылку и многое другое.

Далее следуют разделы, посвященные каскадным таблицам стилей (CSS). С помощью CSS можно задавать точные характеристики практически всех элементов веб-страницы. Это позволяет контролировать внешний вид веб-страницы в окне веб-браузера и приближает возможности веб-дизайна к настольным издательским системам.

Завершает учебное издание раздел, посвященный языку программирования JavaScript. У веб-страниц, созданных с использованием HTML и CSS, есть существенный недостаток – они являются статическими, то есть не могут меняться, реагируя на действия пользователя. Внедрение в HTML программ на языке JavaScript позволит "оживить" веб-страницу, сделать ее интерактивной или, другими словами, заставить взаимодействовать с пользователем. С помощью JavaScript можно обрабатывать данные формы до отправки на сервер, получать информацию о веб-браузере пользователя и его мониторе, и соответствующим образом изменять форматирование страницы, создавать новые окна, изменять любые

элементы HTML-документа в ответ на какое-либо событие, создавать часы на веб-странице, показывающие текущее время с точностью до секунды, скрывать и отображать элементы веб-страницы и многое другое.

Простота и ясность изложения материала позволяет успешно осваивать различные по сложности темы и закреплять пройденный материал, отвечая на вопросы для повторения и самопроверки, которые помещены после каждого раздела. Ответы на вопросы также помогут при подготовке к промежуточному и итоговому контролю знаний по дисциплине. В приложения вынесены пример для оформления отчета по лабораторной работе, таблица цветов и специальных символов, которые помогут как в освоении материала пособия, при подготовке индивидуальных практических заданий, так и в дальнейшей профессиональной деятельности.

Данное издание предназначено, в первую очередь, для студентов специальности "Прикладная лингвистика", изучающих дисциплину "Компьютерные системы перевода и документирования информациии", но может быть использовано и студентами других специальностей для изучения основных аспектов веб-технологий.

1. ИСТОРИЯ СОЗДАНИЯ HTML, CSS, JAVASCRIPT

1.1. Краткая история HTML

Начало официальной истории языка HTML было положено в 1986 году, когда Международной организацией по стандартизации (ISO) был принят стандарт ISO 8879:1986 Information processing – Text and office systems – Standard Generalized Markup Language (SGML). SGML представлял собой обобщенный метаязык, позволяющий строить системы логической, структурной разметки любых разновидностей текстов. При этом управляющие коды не несли никакой информации о внешнем виде документа, а только задавали его логическую структуру. Так, задать размер шрифта в SGML считалось противоречащим стандартам, т. к. не обеспечивало кроссбраузерности и кроссплатформенности представленного в таком виде документа. А основной целью создания каких-либо стандартов было достижение именно этого.

Вследствие таких ограничений, размеченный текст без труда интерпретировался любой компьютерной программой и любым устройством вывода. При этом вид и размер текста задавался исключительно настройками последних. Все это обеспечило небывалую совместимость и универсальность SGML. SGML получил широкое признание и стал активно использоваться в больших проектах. Но в целом этот язык был громоздок и труден для изучения.

Очевидно, что SGML не был готовой системой для разметки текста и не предполагал наличия того или иного списка структурных элементов языка, которые должны применяться в определенных обстоятельствах. Этот язык только подразумевал описание синтаксиса написания основных элементов разметки текста, которые в последующем были названы "тэгами". Для практической же разметки документов нужно было создать язык, который:

1) описывал бы в каких случаях и какой именно элемент языка необходимо применить;

2) давал бы перечень элементов языка, которые могут быть использованы для создания документа и которые должны читаться программами,

работающими с этими документами.

В 1989 году сотрудник Европейской организации по ядерным исследованиям (CERN) Тим Бернерс-Ли (Tim Berners-Lee) выдвинул предложение о создании Системы гипертекстовых документов. В 1990 году он назвал ее World Wide Web (Всемирная паутина). Одной из составляющих системы был язык гипертекстовой разметки. Его созданием Бернерс-Ли занялся в 1990 году, когда разрабатывал первый браузер – программу, позволяющую просматривать гипертекстовые документы.

В 1991 году группой Бернерса-Ли, работавшей над созданием системы передачи гипертекстовой информации через Интернет, было принято решение выбрать SGML в качестве основы для нового языка разметки гипертекстовых документов. Этот язык был назван Hyper Text Markup Language (HTML) (язык гипертекстовой разметки).

HTML разделял все особенности идеологии SGML, т.е. подразумевалась только логическая разметка текста. Например, в HTML версии 1.2 (июнь 1993 г.) присутствовало около 40 тегов. И только три из них отвечали за физические параметры отображения документа (но эти теги не рекомендовались к использованию). В описании одного из этих тегов было сказано: "При просмотре документа, созданного с использованием данного тега текст может отображаться в графических браузерах полужирным курсивом". Первым графическим браузером считается программа Mosaic, разработанная в Национальном центре суперкомпьютерных приложений.

В 1994 году был создан Консорциум W3 (W3C). Вначале он был консультативным органом для лидеров компьютерной индустрии. Крупнейшие мировые компании и корпорации договаривались в W3C об обеспечении совместимости своих продуктов и внедрении новых технологических стандартов. Сейчас консорциум – это организация, разрабатывающая и внедряющая технологические стандарты для Всемирной паутины.

В апреле 1994 г. Консорциумом W3 началась подготовка спецификации следующей версии языка HTML – версии 2.0. Через год был принят окончательный стандарт HTML 2.0, и уже полным ходом шло обсуждение и разработка HTML 3.0. Существенным усовершенствованием HTML 2.0

было введение в язык форм – средств отправки информации от пользователя на сервер.

В том же 1994 г. было решено разбить язык HTML на уровни. Это было сделано, чтобы обеспечить обратную совместимость версий: каждый новый уровень обязательно включал в себя предыдущие. Было выделено четыре уровня:

• Уровень 0 – обязательный для поддержки всеми браузерами. Включает в себя заголовки, ссылки, списки;

• Уровень 1 – добавляются рисунки и элементы начертания текста;

• Уровень 2 – добавляются формы, позволяющие пользователю вводить информацию.

• Уровень 3 – добавляются таблицы, позволяющие размещать на страницах систематизированную определенным образом числовую информацию, а также точно задавать положение объектов на странице.

Например, язык HTML 2.0 является языком второго уровня, поэтому с его помощью нельзя верстать страницы со сложным дизайном, так как это невозможно без использования таблиц.

HTML версии 3.0 считается самым большим прорывом в HTML-технологиях. Первоначальный вариант стандарта включал в себя много интересных нововведений: теги для создания таблиц, разметки математических формул, вставки обтекаемых текстом рисунков, примечаний и др. Но уже тогда появилась потребность в визуальном оформлении гипертекстовых страниц. Не нарушая основ HTML, в W3C решили создать отдельную систему для предоставления возможности описания визуального оформления HTML-документов. Так появились иерархические стилевые спецификации (Cascading Style Sheets, CSS) (каскадные таблицы стилей), имеющие свою структуру, синтаксис и задачи и используемые вместе с HTML.

В 1994 г. группа разработчиков браузера Mosaic под руководством Джеймса Кларка основала корпорацию Netscape Communications и вскоре выпустила первую версию коммерческого браузера Netscape Navigator. Спрос на него при отсутствии альтернативы превысил все ожидания и сделал Netscape Communications к концу 1995 года самой быстрорастущей

компанией в мировой истории.

Чтобы закрепить лидерство Netscape Communications вводила в HTML все новые и новые усовершенствования, которые не были отражены в стандартах W3C и поддерживались только Netscape Navigator. Практически все новые теги были направлены на улучшение внешнего вида документа и расширение возможностей его форматирования. Такая политика компании принесла ей впечатляющий успех. Девять из десяти используемых в то время браузеров были версии Netscape Navigator.

Компания Microsoft изначально не предавала серьезного значения коммерческим перспективам WWW. Однако невероятный взлет Netscape Communications заставил Microsoft изменить свое мнение. В 1996 г. на рынке появился браузер Internet Explorer 2.0, который не получил широкого распространения. Поэтому вскоре появился Internet Explorer 3.0, что разделило рынок браузеров пополам между Navigator Communications и Microsoft. Одновременно с разработкой конкурентного браузера компания Microsoft навела порядок в мире HTML, взяв под свою опеку Консорциум W3. В итоге в сжатые сроки был создан стандарт HTML версии 3.2. Эта версия HTML навела относительный порядок в плане поддержки элементов разметки всеми браузерами.

К концу 1996 года практически все браузеры поддерживали HTML 3.2. Теперь появилась возможность проектировать и отображать на экране сложные композиции графических элементов, ничем не уступающие печатным изданиям. Это положило начало эре веб-дизайна.

В 1997 году появилась спецификация языка HTML 4.0. Она включала поддержку фреймов, унифицированную процедуру вставки различных объектов, поддержку CSS. Кроме того, были усовершенствованы формы и таблицы.

Версия HTML 4.01 стала стандартом в 1999 году. Она также обеспечивает достаточно высокую кроссбраузерность и кроссплатформенность.

На данный момент в разработке находится следующая версия языка HTML – версия 5.0. Ее разработкой занята рабочая группа W3C, в которую входят представители таких компаний, как Microsoft, Opera, Mozilla,

Google, IBM, Apple и другие. Новый стандарт позволит более эффективно управлять мультимедийным содержимым. Также будет улучшена совместимость с новыми языками веб-программирования. И наконец, появятся новые теги, расширяющие возможности веб-дизайна.

Спецификации всех версий HTML можно найти на сайте www.w3c.org.

1.2. Краткая история CSS

История CSS (Cascading Style Sheets) начинается в 1994 году, когда Хокон Виум Ли предложил использовать каскадные таблицы стилей для стилистического оформления веб-страниц. Поскольку язык HTML предполагал все же структурное описание документа, а не визуальное.

В начале 1990-х браузеры имели свои собственные таблицы стилей, HTML тоже кое-что мог предложить в плане визуального оформления. Поэтому CSS не получил широкого распространения.

13 октября 1994 года Марк Андреессен (один из основателей Netscape Communications) сообщил, что первая бета-версия Netscape Navigator доступна для тестирования. За три дня до этого Хокон Виум Ли (норвежский программист, в то время работавший в CERN, теперь сотрудник компании Opera Software) публикует первую черновую версию спецификации CSS, которая имеет мало общего с современными стандартами, но все же общий смысл был заложен именно тогда.

Одним из первых поддержал идею Берт Бос, в то время создававший новый браузер Argo. На основе этого браузера и наработок Хокона Виума Ли происходило дальнейшее развитие CSS. Однако CSS не был единственным языком визуального оформления, который был доступен в то время. Консорциуму W3C было представлено порядка 9 языков стилей.

Как и планировалось, первый черновик CSS был представлен на Веб-конференции в Чикаго в ноябре 1994. Решение технических вопросов и дебаты политического характера продолжались два года, но в конце концов, 17 декабря 1996 года была опубликована первая спецификация CSS и официально рекомендована W3C к использованию. Вся спецификация

была не очень большой, относительно современных версий. Можно было настроить гарнитуру, размер кегля и начертание шрифта; у блочных элементов можно было задавать внешние и внутренние отступы; можно было задавать выравнивание для текста и вписанных в него элементов.

Первым браузером для массового использования, в котором поддерживался новый стандарт, был Microsoft Internet Explorer 3.0, который вышел в августе 1996 года. Этот браузер отлично понимал практически все свойства цвета, фона, шрифта и текста, но использовал блочную модель лишь частично.

Вторым браузером, объявившим о поддержке CSS, был Netscape Navigator 4.0. Разработчики очень скептически относились к CSS, поэтому сделали поддержку больше "для галочки", чтобы не отставать от Microsoft.

Третьим браузером, поддерживающим CSS, стала Opera. Браузер от маленькой, на тот момент никому не известной, норвежской компании получил известность, поскольку был маленьким (помещался на дискету) и хорошо настраивался. Opera версии 3.5, вышедшая в ноябре 1998 г., поддерживала большую часть возможностей CSS1. Этот шаг и дал мощный толчок развитию браузера.

Далее технология стала развиваться быстрыми темпами. 12 мая 1998 г. была принята рекомендация W3C для CSS2, который рекомендовал блочную верстку, позволял использовать механизм селекторов, настраивать курсор, вставлять аудиоэлементы, предоставлял возможность относительного и абсолютного позиционирования и др.

Следует отметить, что спецификация CSS2 с некоторыми изменениями используется до сих пор (CSS2.1 была принята 8 сентября 2009 года). Все браузеры, поддерживающие CSS2.1, также хорошо поддерживают и CSS1.

Сейчас всё еще в стадии разработки находится новая расширенная версия CSS – CSS3, которая будет позволять закруглять уголки блоков, анимировать элементы веб-страницы, делать трехмерные преобразования, хранить переменные и др. И многие современные браузеры уже поддерживают некоторые возможности новой спецификации.

Таким образом, сегодня CSS – это общепринятый стандарт разработки, который принимается/применяется всеми без исключения разработчи-

ками и компаниями, более того, косвенно требуется некоторыми гигантами, например, Google.

1.3. История языка Java Script

Язык JavaScript изобрел Брендан Айк (Brendan Eich) (компания Netscape), перед которым была поставлена задача внедрить язык программирования Scheme или что-то похожее в браузер Netscape. Поскольку требования были размыты, Айка перевели в группу, ответственную за серверные продукты, где он проработал месяц, занимаясь улучшением протокола HTTP. В мае 1995 г. разработчик был переведен обратно в команду, занимающуюся клиентской частью (браузером), где он начал разрабатывать концепцию нового языка программирования. Менеджмент разработки браузера, включая Тома Пакина, Михаэля Тоя, Рика Шелла, был убеждён, что Netscape должен поддерживать язык программирования, встраиваемый в HTML-код страницы.

Помимо Брендона Айка в разработке участвовали один из основателей Netscape Communications Марк Андреессен и один из основателей Sun Microsystems Билл Джой. Чтобы успеть закончить работы над языком к релизу браузера компании заключили соглашение о сотрудничестве в разработке. Они ставили перед собой цель создать "язык для склеивания" составляющих частей веб-ресурса: изображений, плагинов, Java-апплетов, который был бы удобен как для веб-дизайнеров, так и для программистов.

Первоначально язык назывался LiveScript и предназначался как для программирования на стороне клиента, так и для программирования на стороне сервера (там он должен был называться LiveWire). На синтаксис оказали влияние языки Си и Java и поскольку Java в то время было модным словом, 4 декабря 1995 года LiveScript переименовали в JavaScript, получив соответствующую лицензию у Sun. Анонс JavaScript со стороны представителей Netscape и Sun состоялся накануне выпуска второй бета-версии Netscape Navigator. В ней декларируется, что 28 лидирующих ИТ-компаний выразили намерение использовать в своих будущих продуктах JavaScript как объектный скриптовый язык с открытым стандартом.

В 1996 году компания Microsoft выпустила аналог языка JavaScript, названный JScript. Анонсирован этот язык был 18 июля 1996 года. Первым браузером, поддерживающим эту реализацию, был Internet Explorer 3.0.

По инициативе компании Netscape ассоциацией ECMA была проведена стандартизация языка. Развитие этого стандарта началось в ноябре 1996. Первое издание стандарта ECMA было принято Генеральной Ассамблеей ECMA в июне 1997 г. Данный ECMA Стандарт был представлен международной комиссии по стандартам ISO/IEC JTC 1 для принятия, и одобрен как международный эталон ISO/IEC 16262 в апреле 1998 г. Генеральная Ассамблея ECMA в июне 1998 г. одобрила второе издание ECMA-262 с сохранением всех требований ISO/IEC 16262.

Стандартизированная версия имеет название ECMAScript, описывается стандартом ECMA-262. Первой версии спецификации соответствовал JavaScript версии 1.1, а также языки JScript и ScriptEasy.

В настоящее время используется третье издание ECMA-262, которое включает мощные регулярные выражения, лучшую обработку строк, новые инструкции контроля и управления, перехват и обработку исключительных ситуаций, более жесткое определение ошибок, форматирование для числового вывода и незначительные изменения в ожидании ввода средств многоязычности и будущего развития языка.

Работа над языком еще не закончена. Технический комитет работает над существенными расширениями, включая механизмы для сценариев, которые будут созданы для использования в Internet, и более жесткой координацией с другими основными стандартами групп W3C и Wireless Application Protocol форум.

2. ЗНАКОМСТВО С ПРОГРАММОЙ NOTEPAD++

2.1. Установка программы Notepad++

Notepad++ – это редактор, представляющий альтернативу стандартной утилите Notepad, которая входит в поставку Windows. Этот редактор можно использовать как для написания и редактирования программного кода, так и в качестве текстового редактора.

Чтобы установить Notepad++ прежде всего необходимо запустить файл install.exe, выбрать язык установки и нажать на клавишу OK (рис. 2.1).

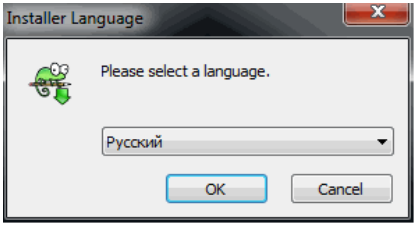


Рис. 2.1. Диалоговое окно выбора языка установки

После этого запускается мастер установки Notepad++ (рис. 2.2) и начинается стандартная установка.

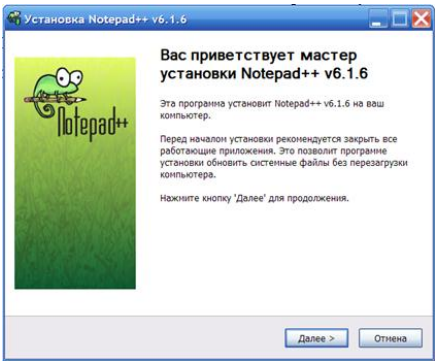


Рис. 2.2. Мастер установки Notepad++

На следующем этапе Notepad++ предлагает окно для выбора папки для установки программы (рис. 2.3)

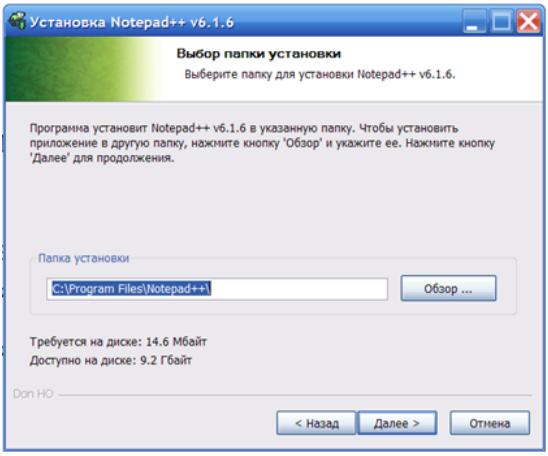


Рис. 2.3. Окно выбора паки для установки

После этого в диалоговом окне установки компонентов программы рекомендуется выбрать все имеющиеся в списке компоненты (рис. 2.4).

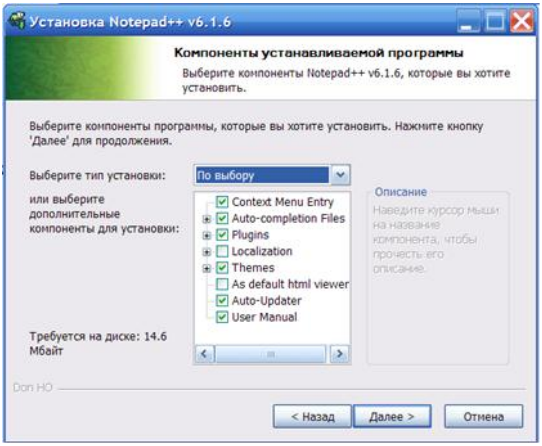


Рис. 2.4. Диалоговое окно установки компонентов программы

В следующем диалоговом окне рекомендуется выбрать один из пунктов (рис. 2.5). Выбор пункта 1 означает, что плагины будут загружаться из инсталлируемой директории на внешнем носителе. Выбор пункта 2 означает, что плагины будут подгружаться из системной папки %APPDATA%.

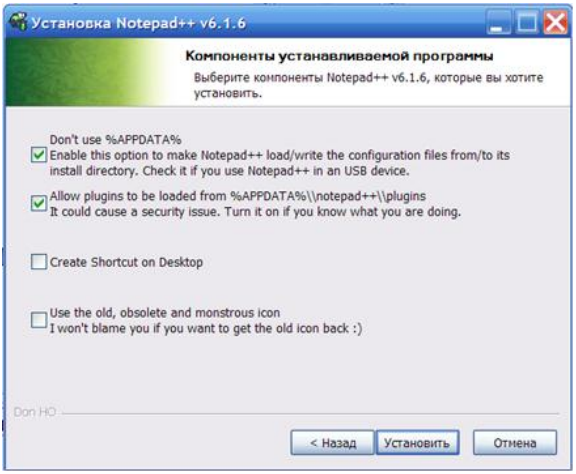


Рис. 2.5. Диалоговое окно установки компонентов программы

После этого следует нажать на кнопку Установить.

2.2. Главное меню Notepad++

После установки программы Notepad++ и настройки ее компонентов программу можно запускать. Стандартное окно программы после первого ее запуска представлено на рис. 2.6.

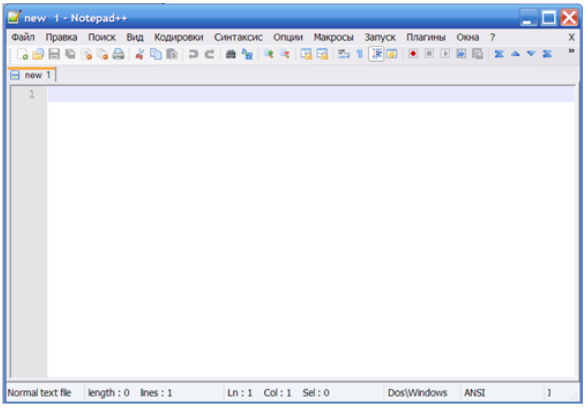


Рис. 2.6. Стандартное окно программы Notepad++

В верхней части окна расположена строка меню Notepad++, состоящая из следующих пунктов: Файл, Правка, Поиск, Вид, Кодировки, Син-

таксис, Опции, Макросы, Запуск, Плагины, Окна.

В меню Файл представлены функции, необходимые для работы с файлами, например, такие как Открыть, Сохранить, Переименовать, Закрыть, Перезагрузить с диска, Удалить с диска, Загрузить сессию, Сохранить сессию и др. (рис. 2.7).

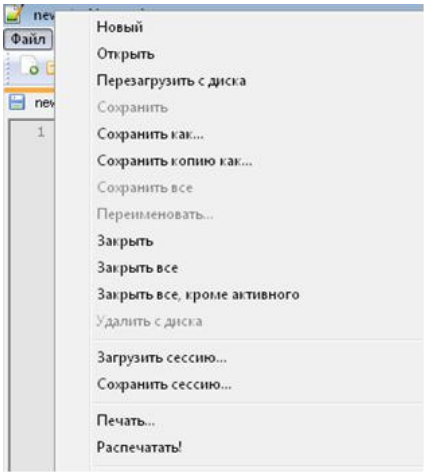


Рис. 2.7. Меню Файл

В меню Правка программы Notepad++ представлены функции для работы с содержимым файлов, позволяющие вырезать, копировать, вставлять, удалять отдельные фрагменты кода, преобразовывать регистр, производить операции со строками (разбиение строк, дублирование, объединение), комментировать строки и блоки кода, устанавливать автоматическое завершение для функций и слов, форматировать конец строки, осуществлять операции с пробелами, делать специальные вставки, выделять блоки столбцом, редактировать столбцы (рис. 2.8.)

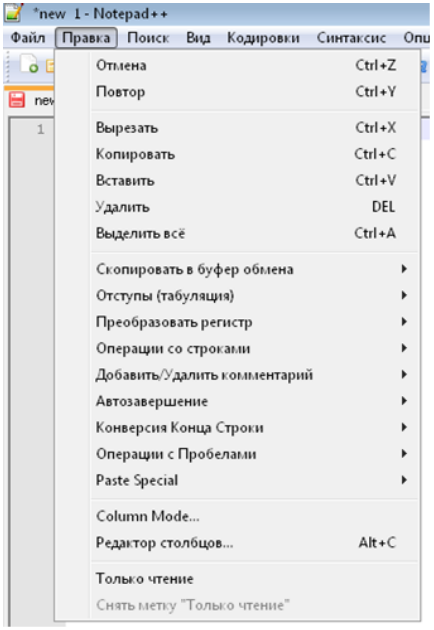


Рис. 2.8. Меню Правка

В меню Поиск программы Notepad++ представлены различные виды функций поиска (рис. 2.9).

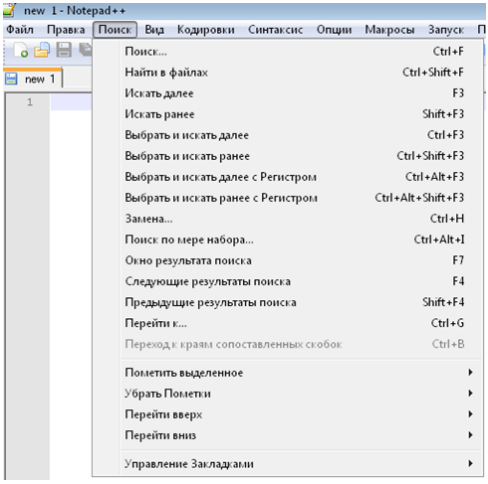


Рис. 2.9. Меню Поиск

Команда Поиск меню Поиск вызывает диалоговое окно с несколькими вкладками, позволяющими осуществлять простой поиск в открытой вкладке с документом, поиск с заменой или поиск во всех документах определенной папки, а также делать пометки или закладки.

В меню Вид представлены функции работы с активным окном, различные операции с блоками, функции синхронизации прокрутки, операции изменения направления текста (рис. 2.10).

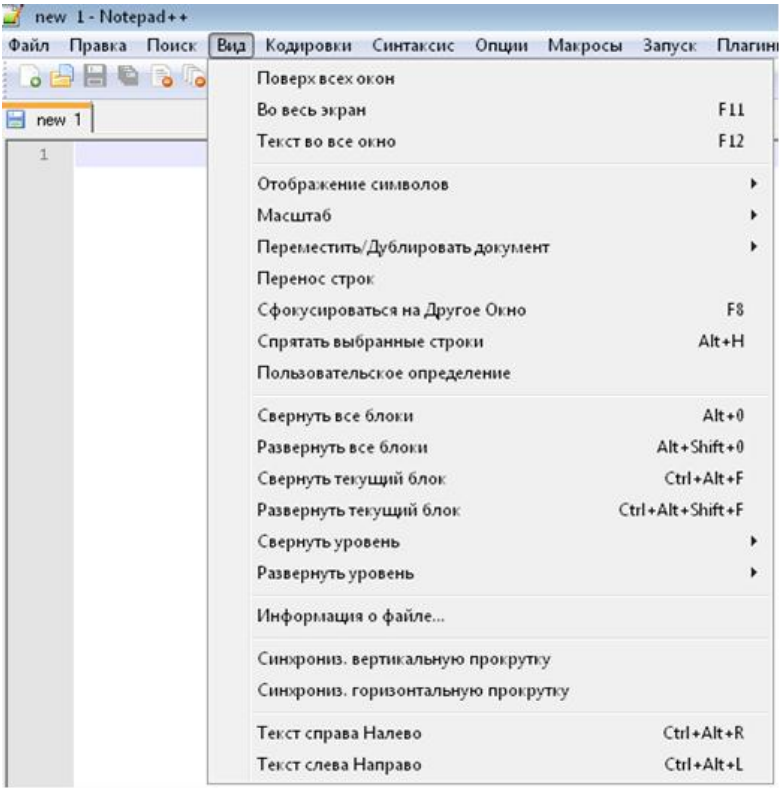


Рис. 2.10. Меню Вид

Команда Информация о файле меню Вид вызывает окно, содержащее различную информацию об открытом файле: путь к файлу, дату создания, дату внесения изменений, размер файла, количество символов (без пробелов), количество слов, строк и др. (рис. 2.11).

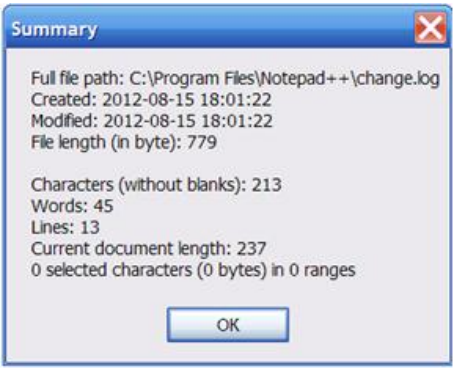


Рис. 2.11. Окно, содержащее информацию о файле

В меню Кодировки представлены функции кодировки файлов, а также функции преобразования кодировок из одного вида в другой (рис. 2.12).

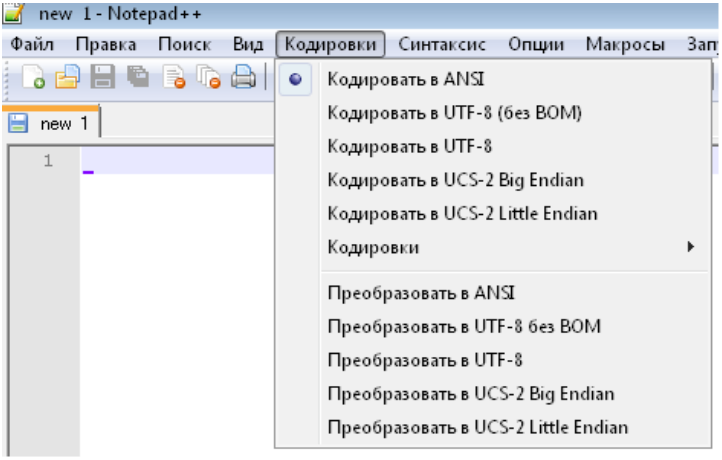


Рис. 2.12. Меню Кодировка

В Notepad++ поддерживается работа с кодировками ANSI, UTF-8, UCS-2 в формате MAC, Linux, Windows. Текущая кодировка открытого файла отображается в строке состояния (рис. 2.13).

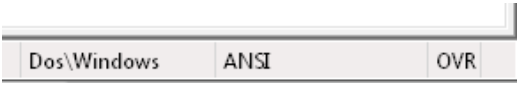


Рис. 2.13. Отображение кодировки файла

Меню Синтаксис дает пользователю возможность воспользоваться

функцией подсветки синтаксиса, которая поддерживается для более чем 40 различных языков программирования, например, таких как C, C#, C++, CSS, HTML, Java, Javascript, Pascal, Perl, PHP, Python, XML (рис. 2.14).

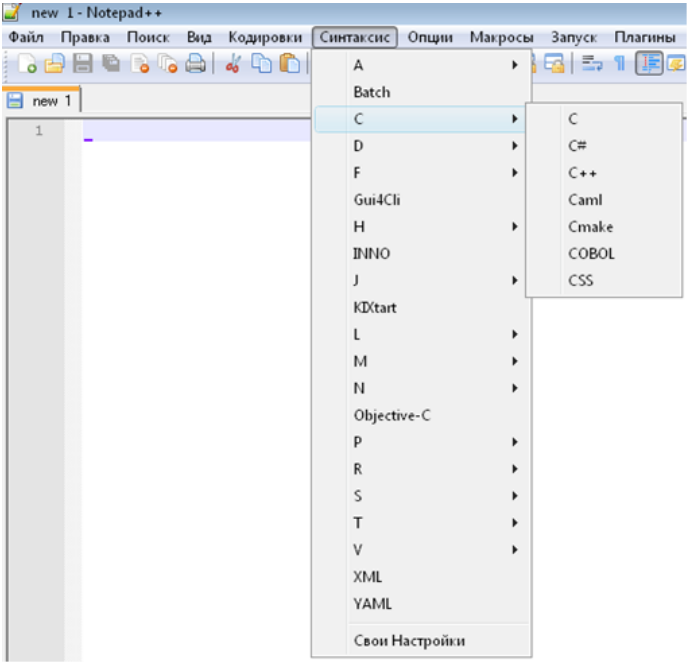


Рис. 2.14. Меню Синтаксис

Благодаря подсветке синтаксиса в Notepad++ наглядно видно имеются ли ошибки в программном коде, поскольку цветовое оформление кода в месте, где была допущена ошибка, будет изменено по сравнению со стандартным цветовым оформлением.

Стиль подсветки по умолчанию выбирается в зависимости от расширения файлов, но пользователь может и вручную настроить эту опцию. Для этого необходимо в главном меню Notepad++ выбрать пункт Опции и команду Определение стилей, а затем в открывшемся диалоговом окне вручную установить необходимые настройки (рис. 2.15).

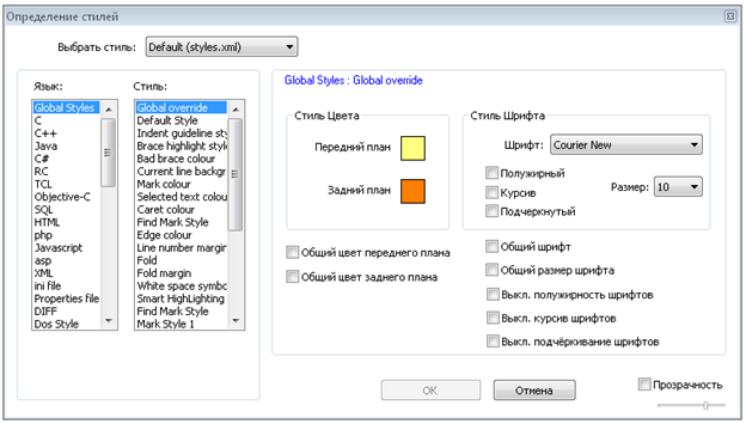


Рис. 2.15. Окно пользовательской настройки подсветки синтаксиса

Меню Опции, кроме упомянутой выше функции настройки подсветки синтаксиса, содержит функции редактирования контекстного меню, импорта плагинов и тем, работы с горячими клавишами, а также позволяет устанавливать и изменять основные настройки программы Notepad++ (рис. 2.16).

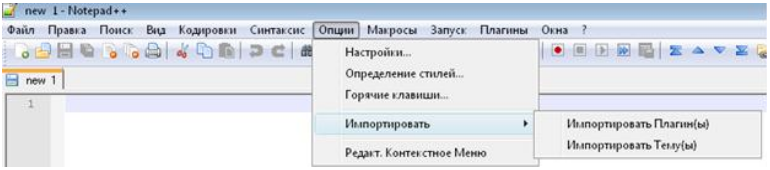


Рис. 2.16. Меню Опции

Меню Макросы позволяет пользователю создавать, воспроизводить, удалять макросы, а также назначать и изменять для них горячие клавиши (рис. 2.17).

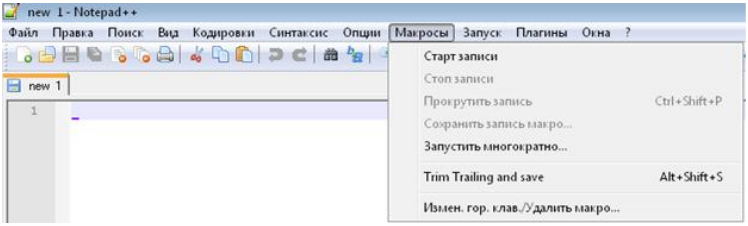


Рис. 2.17. Меню Макросы

Макросы позволяют полностью автоматизировать задачи, которые со-

стоят в многократном повторении определенных операций или действий.

Команды меню Запуск позволяют: просмотреть результаты работы пользователя в различных браузерах (Firefox, Internet Explorer, Google Chrome, Safari), открыть файл, открыть папку, содержащую файл, отправить файл через Outlook Express (рис. 2.18).

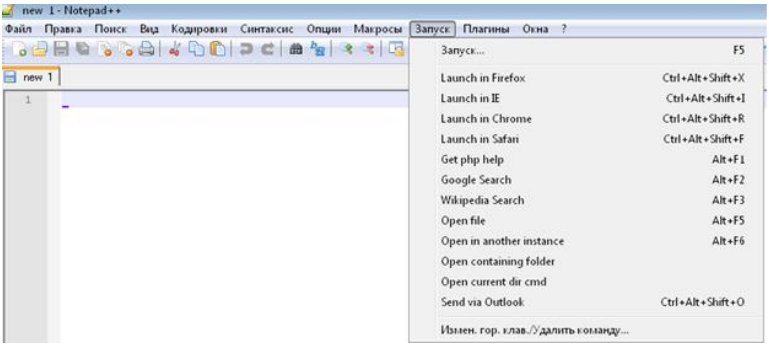


Рис. 2.18. Меню Запуск

Для получения описания php кода из файла пользователя можно воспользоваться командой Get php help, которая автоматически переадресовывает пользователя на сайт http://php.net/, где он и может получить это описание.

Для того, чтобы найти необходимую информацию в поисковой системе Google или в свободной энциклопедии Wikipedia можно воспользоваться командами Google Search и Wikipedia Search соответственно. Эти команды автоматически переадресовывают пользователя на соответствующие сайты, на которых уже представлены результаты поиска.

Меню Плагины содержит перечень плагинов, доступных по умолчанию в Notepad++ (рис. 2.19).

Рис. 2.19. Меню Плагины

Другие плагины можно скачать на официальном сайте Notepad++ и

установить с помощью встроенного в Notepad++ плагин-менеджера.

Плагин Compare используется для сравнения разных версий одного и того же файла, а также для сравнения двух файлов, при его вызове будут подсвечиваться элементы, которые отсутствуют/присутствуют в одной из версий файла, и наоборот, либо будут подсвечиваться существующие различия в сравниваемых файлах (рис. 2.20).

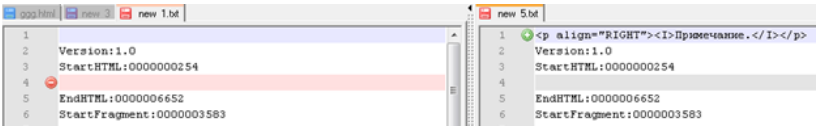


Рис. 2.20. Сравнение двух документов

Для активации режима сравнения кроме команды Compare меню Плагины можно использовать сочетание клавиш Alt+D. В результате в рабочем окне Notepad++ будет открыта активная в данный момент вкладка и вкладка, расположенная рядом с ней. Два документа в окне редактора Notepad++ можно просматривать как в вертикальном, так и в горизонтальном положении. Для выбора способа отображения документов нужно подвести курсор к границе разделения документов, щелкнуть правой кнопкой мыши и выбрать из контекстного меню одну из команд Rotate to left (повернуть влево) или Rotate to right (повернуть вправо). Для перехода в обычный режим работы с документами нужно в меню Плагины выбрать команду Compare → Clear Results или использовать сочетание клавиш Ctrl+Alt+D.

Плагин Converter используется для преобразования символов кодировки ASCII в HEX (шестнадцатеричную систему счисления) и наоборот. Команда Conversion panel открывает диалоговое окно Conversion, в котором можно получить сведения о любом введенном пользователем символе в кодировке ASCII и различных системах счисления: десятичной, шестнадцатеричной, бинарной, восьмеричной.

Плагин NppExport используется для экспорта файлов в форматы rtf и html.

Плагин NppFTP позволяет работать с файлами пользователя непосредственно на сервере. С его помощью можно загружать файлы с сервера, вносить в них изменения прямо в окне редактора, и сохраняя файлы, автоматически отправлять их обратно на сервер.

Плагин Spell-Checker предназначен для проверки орфографии.

Команда Plugin Manager позволяет осуществлять работу с доступными в Notepad++ и новыми плагинами, загружая, обновляя, переустанавливая и удаляя их.

Меню Окна содержит перечень открытых на данный момент окон, а также команду Окна…, которая позволяет выполнять различные операции с окнами: открывать, сохранять, закрывать, упорядочивать.

2.3. Возможности программы Notepad++

Кроме доступных через главное меню функций программы Notepad++, работу пользователя существенно облегчают и другие ее функции, например такие как, сворачивание кода, выделение парных скобок, автоматическое завершение слов, одновременная работа с несколькими документами, установка заметок и др.

Сворачивание кода – это возможность "скрывать" фрагменты кода, расположенные между скобками тега в одной функции, между операторами выбора или цикла. Для того чтобы свернуть фрагмент кода, находящийся между тегами, необходимо нажать на знак  , расположенный в левой части рабочего окна программы, таким образом можно определить в каком месте кода находится закрывающий тег. На рисунке 2.21 представлен результат операции сворачивания фрагмента кода.

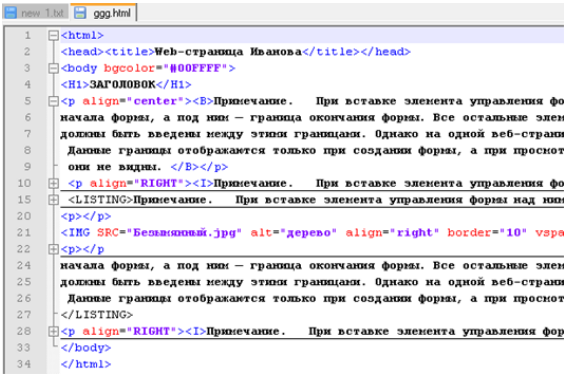


Рис. 2.21. Внешний вид рабочего окна программы Notepad++ после сворачивания фрагмента кода

Функция выделения (подсветки) парных скобок/тегов программы Notepad++ позволяет наглядно отобразить структуру кода (рис. 2.22), что способствует нахождению допущенных в программном коде ошибок, например, при отсутствии закрывающего тега, или позволяет разделить код на отдельные сегменты.

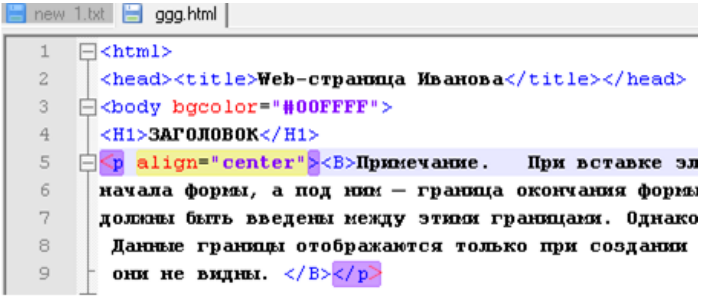


Рис. 2.22. Подсветка парных скобок/тегов программы Notepad++

Функция автоматического завершения слов позволяет по первым набранным пользователем символам выбрать необходимый вариант завершения слова из предлагаемого набора слов, характерного для определенного языка программирования. Для активации этой функции необходимо после ввода первых символов слова воспользоваться сочетанием клавиш Ctrl+Пробел, и затем в появившемся окне выбрать нужный вариант слова (рис. 2.23).

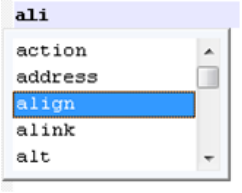


Рис. 2.23. Окно автоматического завершения слова

Чтобы сделать автозавершение функций и слов автоматическим (без использования горячих клавиш) следует воспользоваться командой Настройки меню Опции, а затем в диалоговом окне Настройка на вкладке Резерв/Автозавершение установить опцию Включить для каждого ввода.

Программа Notepad++ предоставляет пользователю возможность одновременной работы с несколькими файлами. Открываются эти файлы не

в разных окнах, а в разных вкладках рабочего окна программы. Эти вкладки можно произвольно перетаскивать мышью с места на место, закрывать, сворачивать. При закрытии окна программы соответственно закроются и все файлы, открытые на тот момент, однако, при очередном запуске программы все закрытые вкладки будут восстановлены (по аналогии с работой некоторых браузеров).

Редактор Notepad++ постоянно отслеживает состояние открытых в нем файлов и в случае их редактирования или удаления, например, в другой программе, пользователю будет предложено либо закрыть данную вкладку с файлом, либо повторно подгрузить файл в Notepad++.

В Notepad++ можно одновременно работать с двумя копиями одного и того же документа. Для активации этого режима необходимо щелкнуть правой кнопкой мыши по названию нужной вкладки, а затем из контекстного меню выбрать команду Дублировать в другое окно. Изменения, внесенные в одну из копий документа, автоматически вносятся и в другую копию.

При работе с программным кодом в окне Notepad++ пользователю предоставляется возможность делать заметки. Заметка имеет вид синей точки в левой части рабочего окна программы (рис. 2.24).

Поставить заметку можно либо с помощью мыши, щелкнув левой кнопкой мыши справа от номера строки, либо с использованием сочетания клавиш Ctrl+F2. Чтобы переместиться к следующей заметке, необходимо воспользоваться клавишей F2, а для перемещения к предыдущей заметке – сочетанием клавиш Shift+F2.

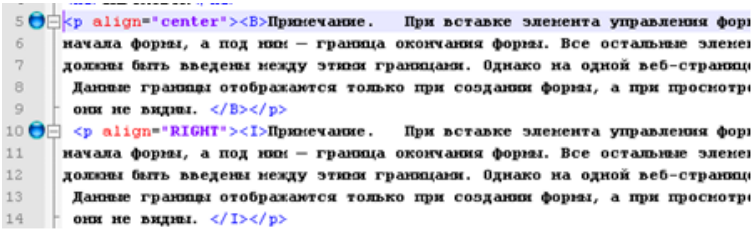


Рис. 2.24. Окно с заметками

Для открытия файла в Notepad++ можно воспользоваться командой Открыть меню Файл или кнопкой Открыть на панели задач, либо просто

перетащить ярлык документа прямо в рабочее окно программы.

Несмотря на все многообразие и разнообразие предоставляемых функций, Notepad++ не ограничивается встроенными возможностями. Для расширения функциональности можно скачать на официальном сайте Notepad++ дополнительные плагины, можно добавить файлы поддержки автоматического завершения слов, например, для языков программирования которые не входят в стандартную поставку.

Больше всего программа Notepad++ подходит для быстрого редактирования исходных кодов, особенно веб-приложений и скриптовых языков, где не требуется перекомпиляция проектов.

Практические задания

Ознакомиться с функциональными возможностями программы Notepad++. С помощью текстовых файлов освоить ее основные функции.

Контрольные вопросы

1. Перечислите основные функции, которые выполняет текстовый редактор Notepad++.
2. Что такое плагин? Какие плагины можно использовать для программы Notepad++?
3. Для каких языков программирования и языков разметки можно использовать текстовый редактор Notepad++?
4. Что общего между текстовым редактором Notepad++ и MS Word?
5. Сколько основных плагинов существует на данный момент для программы Notepad++? Назовите их.
6. В чем основные преимущества программы Notepad++ перед другими программами?

3. ВВЕДЕНИЕ В HTML

«Дизайн – это не только то, как это выглядит и как чувствуется. Дизайн – это еще и то, как это работает.» – Стив Джобс

HTML (от англ. HyperText Markup Language – «язык разметки гипертекста») – стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц создаются при помощи языка HTML (или XHTML). Язык HTML интерпретируется браузерами и отображается в виде документа в удобной для человека форме. HTML является приложением («частным случаем») SGML (стандартного обобщённого языка разметки) и соответствует международному стандарту ISO 8879. XHTML же является приложением XML.

3.1. Структура HTML-документа

Структура, характерная для любого HTML-документа, представлена ниже.

```
<!DOCTYPE> <!-- Объявление формата документа -->
<html>
<head>
<!-- Техническая информация о документе -->
</head>
<body>
<!-- Основная часть документа -->
</body>
</html>
```

Тег <!DOCTYPE> позволяет определить веб-браузеру формат файла и правильно отобразить все его инструкции. Допустимые форматы для HTML 4.01:

1. Strict – строгий формат. Не содержит тегов и параметров, помеченных как устаревшие или не одобряемые. Объявление формата:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
```

"http://www.w3.org/TR/html4/strict.dtd">

2. Transitional – переходный формат. Содержит устаревшие теги для обеспечения совместимости и упрощения перехода с предыдущих версий HTML. Объявление формата:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

3. Frameset – аналогичен переходному формату, но содержит также теги для создания фреймов. Объявление формата:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">

Если тег <!DOCTYPE> не указан, то веб-браузеры могут по-разному отображать веб-страницу.

Весь текст HTML-документа следует располагать между тегами <html> и </html>. HTML-документ состоит из двух разделов – заголовка (между тегами <head> и </head>) и содержательной части (между тегами <body> и </body>).

3.2. Раздел HEAD. Техническая информация о документе

Раздел HEAD содержит техническую информацию о странице: заголовок, описание страницы, ключевые слова для поисковых машин, данные об авторе и времени создания страницы, о базовом адресе страницы, кодировке и т. д.

Единственным обязательным тегом в разделе HEAD является тег <title>. Текст, расположенный между тегами <title> и </title>, отображается в строке заголовка веб-браузера. Длина заголовка должна быть не более 60 символов: *<title>Заголовок страницы</title>*

Очень часто текст между тегами <title> и </title> используется в результатах, выдаваемых поисковыми системами, в качестве текста ссылки на эту страницу. Поэтому заголовок должен максимально полно описывать содержание страницы.

С помощью одинарного тега <meta> можно задать описание содержимого страницы и ключевые слова для поисковых машин. Если текст

между тегами <title> и </title> используется в качестве текста ссылки на эту страницу, то описание из тега <meta> будет отображено под ссылкой:

<meta name="description" content="Описание содержимого страницы">

<meta name="keywords" content="Ключевые слова через запятую">

Можно также указать несколько описаний на разных языках. Для этого в параметре lang следует указать используемый язык:

<meta name="description" lang="ru" content="Описание содержимого страницы">

<meta name="description" lang="en" content="Description">

<meta name="keywords" lang="ru" content="Ключевые слова через запятую">

<meta name="keywords" lang="en" content="Keywords">

Кроме того, с помощью тега <meta> можно запретить или разрешить индексацию веб-страницы поисковыми машинами:

<meta name="robots" content="<Индексация>, <Переход по ссылкам>">

Параметр content может принимать следующие значения:

- index – индексация разрешена;
- noindex – индексация запрещена;
- follow – разрешено переходить по ссылкам, которые находятся на этой веб-странице;
- nofollow – запрещено переходить по ссылкам;
- all – комбинация index плюс follow;
- none – комбинация noindex плюс nofollow.

Приведем несколько примеров.

Индексация и переход по ссылкам разрешены:

<meta name="robots" content="index, follow">

Индексация разрешена, а переход по ссылкам запрещен:

<meta name="robots" content="index, nofollow">

Индексация и переход по ссылкам запрещены:

<meta name="robots" content="noindex, nofollow">

Также с помощью тега <meta> можно указать кодировку текста:

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">

Для автоматической перезагрузки страницы через заданный проме-

жуток времени следует воспользоваться свойством refresh тега <meta>:
<meta http-equiv="refresh" content="30">

В примере, приведенном выше, страница будет перезагружена через 30 секунд.

Если сразу после перезагрузки необходимо переадресовать посетителя на другую страницу, то можно указать URL-адрес в параметре url:
<meta http-equiv="refresh" content="0; url=http://mail.ru/">

В разделе HEAD могут быть расположены также теги <base>, <link>, <script>, <style> и некоторые другие.

3.3. Раздел BODY. Основная часть документа

В этом разделе располагается все содержимое документа, поэтому большинство тегов должны находиться именно между тегами <body> и </body>. Следует отметить, что в формате Strict содержимое тега <body> должно быть расположено внутри блочных элементов, например, , или др.:

<body>
* Текст документа *
* Текст документа *
</body>

Тег <body> имеет следующие параметры:

- bgcolor задает цвет фона веб-страницы.

Цвет задается в шестнадцатеричной кодировке. Для каждой составляющей цвета (красного, зеленого и синего) задается значение в пределах от 00 до FF. Эти значения объединяются в одно число, перед которым добавляется символ "#", например, значение #FF0000 соответствует красному цвету, #00FF00 – ярко-зеленому, а #FF00FF – фиолетовому;

- background позволяет задать фоновый рисунок для документа путем указания URL-адреса изображения;
- alink определяет цвет активной ссылки;
- link устанавливает цвет еще не просмотренных ссылок;
- vlink определяет цвет уже просмотренных ссылок;

- text устанавливает цвет текста.

Например, тег <body> может выглядеть так:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Заголовок страницы</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body alink="#FF0000" link="#000000" vlink="#000080" text="#000000">
Текст документа
</body>
</html>

3.4. Выделение фрагментов текста

Тег отображает текст полужирным шрифтом: *Полужирный шрифт*

Вместо тега лучше использовать тег логического форматирования : *Полужирный шрифт*

Тег отображает текст курсивом: *Текст, выделенный курсивом*

Вместо тега лучше использовать тег логического форматирования : *Текст, выделенный курсивом*

Тег <u> отображает текст подчеркнутым: *<u>Подчеркнутый текст</u>*

Теги <strike> и <s> отображают текст перечеркнутым:

<strike>Перечеркнутый текст</strike>
<s>Перечеркнутый текст</s>

3.5. Создание нижних и верхних индексов

Тег <sub> сдвигает текст ниже уровня строки и уменьшает размер шрифта. Он используется для создания нижних индексов. Например, H_2O: *Формула воды H₂O*

Тег <sup> сдвигает текст выше уровня строки и уменьшает размер шрифта. Этот тег используется для создания верхних индексов, например, $м^2$:
Единица измерения площади – м²

3.6. Вывод текста заданным шрифтом

Тег определяет тип, размер и цвет шрифта. Он имеет следующие параметры:

1) face служит для указания типа шрифта:

Текст

Можно указать как один, так и несколько типов, разделяя их запятыми.

При этом список шрифтов просматривается слева направо. Указанное название должно точно соответствовать названию типа шрифта. Если указанный шрифт не найден на компьютере пользователя, то используется шрифт по умолчанию;

2) size задает размер шрифта в условных единицах от 1 до 7. Размер, используемый веб-браузером по умолчанию, принято приравнивать к 3.

Размер шрифта можно указывать как цифрой от 1 до 7, так и в относительных единицах, указывая, на сколько единиц нужно увеличить (знак "+") или уменьшить (знак "–") размер шрифта по сравнению с базовым:

Текст
Текст
Текст

3) color позволяет указывать цвет шрифта. Цвета задаются так же, как для параметра bgcolor, т.е. в шестнадцатеричной кодировке:

Текст

Вместо цифр можно использовать названия цветов на английском языке:

Текст

Чаще всего используются такие цвета:

- black – #000000 – черный;
- white – #FFFFFF – белый;
- yellow – #FFFF00 – желтый;
- silver – #C0C0C0 – серый;

- red – #FF0000 – красный;
- green – #008000 – зеленый;
- gray – #808080 – темно-серый;
- blue – #0000FF – синий;
- purple – #800080 – фиолетовый.

Также для форматирования текста применяются и другие теги. Для вывода текста шрифтом большего размера используется парный тег <big>:

Текст <big>большего</big> размера

А для вывода текста шрифтом меньшего размера применяется парный тег <small>:

Текст <small>меньшего</small> размера

Для вывода текста моноширинным шрифтом используется тег <tt>:

<tt>Моноширинный шрифт</tt>

3.7. Форматирование документа

Практически все теги, рассмотренные в предыдущем разделе, являются тегами физического форматирования. Исключение составляют теги и . Эти теги являются тегами логического форматирования текста и используются для выделения очень важных и просто важных фрагментов соответственно. Теги логического форматирования используются для структурной разметки документа и могут отображаться разными веб-браузерами по-разному. Перечислим основные теги логического форматирования:

- <cite>…</cite> – отмечает цитаты и названия произведений;
- <code>…</code> – отмечает фрагменты программного кода;
- <acronym>…</acronym> – отмечает аббревиатуры;
- <kbd>…</kbd> – отмечает фрагмент как вводимый пользователем с клавиатуры;
- <q>…</q> – отмечает короткие цитаты;
- <samp>…</samp> – отмечает результат, выдаваемый программой;
- <var>…</var> – отмечает имена переменных.

3.8. Заголовки

Заголовки могут иметь шесть различных размеров:

<hx>Заголовок</hx>

где x – число от 1 до 6.

Заголовок с номером 1 является самым крупным:

<h1>Самый крупный заголовок</h1>

Заголовок с номером 6 является самым мелким:

<h6>Самый мелкий заголовок</h6>

Основным параметром тега <hx> является параметр align, который задает выравнивание заголовка относительно окна веб-браузера. Параметр align может принимать следующие значения:

– center – выравнивание по центру:

<h1 align="center">Заголовок первого уровня с выравниванием по центру</h1>

– left – выравнивание по левому краю (по умолчанию):

<h2 align="left">Заголовок второго уровня с выравниванием по левому краю</h2>

– right – выравнивание по правому краю:

<h6 align="right"> Заголовок шестого уровня с выравниванием по правому краю</h6>

3.9. Перевод строки

Для разделения строк используется одинарный тег
.

Если в HTML-документе набрать текст:

Строка1
Строка2
Строка3,

то веб-браузер отобразит его в одну строку: "Строка1 Строка2 Строка3".

Для того чтобы строки располагались друг под другом, необходимо добавить тег
 в конец каждой строки:

*Строка1
*
*Строка2
*
*Строка3
*

Для вывода текста в том же виде, что и в исходном коде, можно воспользоваться парным тегом <pre>:

<pre>
Строка1
Строка2
Строка3
</pre>

В этом примере строки также будут располагаться друг под другом.

3.10. Горизонтальная линия

Одинарный тег <hr> позволяет провести горизонтальную линию.

Тег <hr> имеет следующие параметры:

1) size – толщина линии: *<hr size="5">*

2) width – длина линии. Можно указывать значение как в пикселах, так и в процентах относительно ширины окна веб-браузера:

<hr size="5" width="100">
<hr size="5" width="100%">

3) align – выравнивание линии. Параметр может принимать следующие значения:

- center – выравнивание по центру (значение по умолчанию):

<hr size="2" width="200" color="red" align="center">

- left – выравнивание по левому краю:

<hr size="2" width="200" color="red" align="left">

- right – выравнивание по правому краю:

<hr size="2" width="200" color="red" align="right">

- noshade – присутствие этого параметра отменяет рельефность линии:

<hr size="2" width="200" align="center" noshade>

3.11. Тег комментария

Текст, заключенный между тегами <!-- и -->, не отображается веб-браузером. Заметим, что это нестандартная пара тегов, так как открывающий тег не имеет закрывающей угловой скобки, а в закрывающем теге

отсутствует открывающая угловая скобка: *<!-- Текст комментария -->*.

3.12. Бегущая строка. Тег <marquee>

Тег <marquee> создает бегущую строку на странице. Содержимое контейнера <marquee> не ограничивается строками и позволяет перемещать (скролировать) любые элементы веб-страницы – изображения, текст, таблицы, элементы форм и т.д. Перемещение можно задать не только по горизонтали, но и вертикали, в этом случае указываются размеры области, в которой будет происходить движение. Первоначально тег <marquee> был предназначен только для браузера Internet Explorer, но современные версии других браузеров также понимают и поддерживают этот тег.

Атрибуты тега <marquee>:

- behavior. Задает тип движения содержимого контейнера <marquee>.
- bgcolor. Задает цвет фона.
- direction. Указывает направление движения содержимого контейнера <marquee>.
- height. Задает высоту области прокрутки.
- hspace. Задает горизонтальные поля вокруг контента.
- loop. Задает, сколько раз будет прокручиваться содержимое.
- scrollamount. Задает скорость движения контента.
- scrolldelay. Величина задержки в миллисекундах между движениями.
- truespeed. Отменяет встроенный ограничитель скорости при низких значениях атрибута scrolldelay.
- vspace. Задает вертикальные поля вокруг содержимого.
- width. Задает ширину области прокрутки.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Тег MARQUEE</title>
</head>
```

<body>
<marquee behavior="alternate" direction="left" bgcolor="#ffcc00">
Приветствует Вас на нашем сайте !!!
</marquee>
</body>
</html>

3.13. Разделение на абзацы

Тег позволяет разбить текст на отдельные абзацы. Веб-браузеры отделяют абзацы друг от друга пустой строкой. Закрывающий тег необязателен. Основным параметром тега является параметр align, который задает горизонтальное выравнивание. Параметр может принимать такие значения:

- center – выравнивание по центру:

* Абзац с выравниванием по центру *

- left – выравнивание по левому краю (по умолчанию):

* Абзац с выравниванием по левому краю *

- right – выравнивание по правому краю:

* Абзац с выравниванием по правому краю *

- justify – выравнивание по ширине:

* Абзац с выравниванием по ширине *

3.14. Теги и . Группировка элементов страницы

Теги и визуально ничего не делают. Зато они позволяют сгруппировать несколько элементов страницы в один. Кроме того, тег используется для блочной верстки веб-страницы. Если необходимо выделить фрагмент текста внутри абзаца, то следует использовать тег , так как тег отобразит фрагмент на новой строке, а не внутри абзаца. Тег позволяет не только группировать элементы, но и управлять горизонтальным выравниванием с помощью параметра align. Параметр может принимать следующие значения:

- center — выравнивание по центру:

* Элемент с выравниванием по центру *

• left — выравнивание по левому краю:

* Элемент с выравниванием по левому краю *

• right — выравнивание по правому краю:

* Элемент с выравниванием по правому краю *

• justify — выравнивание по ширине (по двум сторонам):

* Элемент с выравниванием по ширине *

3.15. Списки

Список – это набор упорядоченных абзацев текста, помеченных специальными значками (маркированные списки) или цифрами (нумерованные списки).

Маркированный список помещают внутри пары тегов <ul> и . Перед каждым пунктом списка необходимо поместить тег <li>. Закрывающий тег не обязателен. Ниже представлена структура маркированного списка.

<ul>
*
Первый пункт
*
*
Второй пункт
*
*
*

Тег <ul> имеет параметр type, позволяющий задать маркер, которым будут помечены строки списка. Параметр может принимать следующие значения:

• disc – маркер в форме круга с заливкой:

<ul type="disc">
*
Первый пункт
*
*
Второй пункт
*
*
*

• circle – маркер в форме круга без заливки:

<ul type="circle">
*
Первый пункт
*
*
Второй пункт
*
*
*

- square – маркер в форме квадрата с заливкой:

```
<ul type="square">
<li>Первый пункт</li>
<li>Второй пункт</li>
</ul>
```

Нумерованный список помещают внутри пары тегов <ol> и . Перед каждым пунктом списка необходимо поместить тег <li>. Закрывающий тег необязателен.

Пример нумерованного списка:

```
<ol>
<li>Первый пункт</li>
<li>Второй пункт</li>
</ol>
```

Тег <ol> имеет два параметра: type и start.

Параметр type позволяет задать формат, которым будут пронумерованы строки списка. Параметр type может принимать следующие значения:

- A – пункты нумеруются прописными латинскими буквами:

```
<ol type="A">
<li>Первый пункт</li>
<li>Второй пункт</li>
</ol>
```

- a – пункты нумеруются строчными латинскими буквами:

```
<ol type="a">
<li>Первый пункт</li>
<li>Второй пункт</li>
</ol>
```

- I – пункты нумеруются прописными римскими цифрами:

```
<ol type="I">
<li>Первый пункт</li>
<li>Второй пункт</li>
</ol>
```

- i – пункты нумеруются строчными римскими цифрами:

```
<ol type="i">
<li>Первый пункт</li>
<li>Второй пункт</li>
</ol>
```

- 1 – пункты нумеруются арабскими цифрами (по умолчанию):

```
<ol type="1">
<li>Первый пункт</li>
<li>Второй пункт</li>
</ol>
```

Параметр start задает номер, с которого будет начинаться нумерация строк списка:

```
<ol type="1" start="5">
<li>Пятый пункт</li>
<li>Шестой пункт</li>
</ol>
```

Тег <li> также имеет параметр value, который позволяет изменить номер данного элемента списка:

```
<ol type="1">
<li>Первый пункт</li>
<li value="5">Пятый пункт</li>
<li>Шестой пункт</li>
</ol>
```

3.16. Списки определений

Списки определений состоят из пар термин+определение. Они описываются с помощью тега <dl>. Для вставки термина применяется тег <dt>, а для вставки определения тег <dd>. Закрывающие теги </dt> и </dd> необязательны.

Пример списка определений:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
```

<head>
<title>Списки определений</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<dl>
<dt>HTML (HyperText Markup Language)</dt>
<dd>

Язык разметки документа, описывающий форму отображения информации на экране компьютера:

</dd>
<dt>CSS (Cascading Style Sheets)</dt>
<dd>Каскадные таблицы стилей</dd>
</dl>
</body>
</html>

3.17. Графика

Применение графики делает веб-страницу визуально привлекательнее. Изображения помогают лучше передать суть и содержание документа. В Интернете применяются следующие графические форматы:

1) GIF – использует только 256 цветов и поддерживает прозрачность. Кроме того, GIF-файл может содержать анимацию;

2) JPEG – метод сжатия фотографий с потерей качества. Прозрачность и анимация не поддерживаются;

3) PNG – формат хранения графики, использующий сжатие без потерь. Поддерживает прозрачность. Разрабатывался в качестве замены формата GIF.

Однако следует помнить, что перенасыщение документа графикой приводит к увеличению времени загрузки веб-страницы, поэтому желательно применять графику только там, где это действительно оправданно.

3.18. Изображение на веб-странице

Изображения вставляются в веб-страницы с помощью одинарного тега <img>. Сам тег <img> должен быть расположен внутри блочного тега, например, <p>, или др. Тег <img> имеет следующие параметры:

• src – URL-адрес файла графического изображения:

<img src="foto.gif">

<img src="http://www.mysite.ru/foto.gif">

• alt – строка текста, которая будет выводиться на месте появления изображения до его загрузки или при отключенной графике, а также в случае, если изображение загрузить не удалось. Кроме того, при наведении курсора мыши на изображение текст, указанный в параметре alt, можно увидеть в качестве текста всплывающей подсказки:

<img src="foto.gif" alt="Текст подсказки">

• width – ширина изображения в пикселах:

<img src="foto.gif" width="480" alt="Текст подсказки">

• height – высота изображения в пикселах:

<img src="foto.gif" width="480" height="60" alt="Подсказка">

Значения параметров width и height могут не соответствовать реальным размерам изображения. В этом случае веб-браузер выполнит перемасштабирование. Если значение одного из параметров указать неправильно, то изображение будет искажено. Если указать только один параметр, то значение второго будет рассчитано пропорционально значению первого, исходя из реальных размеров изображения.

Всегда указывайте значения параметров width и height, так как это позволит веб-браузеру отформатировать веб-страницу до загрузки изображений. В противном случае загрузка каждого изображения приведет к необходимости произвести форматирование еще раз, что в свою очередь приведет к перемещению других элементов веб-страницы.

Параметры элемента img:

• border – толщина границы изображения:

<img src="foto.gif" border="0" alt="Текст подсказки">

• align – расположение изображения относительно текста или других

элементов веб-страницы. Параметр может принимать следующие значения:

• left – изображение выравнивается по левому краю, а текст обтекает его с правой стороны:

* <img src="foto.gif" align="left" alt="Подсказка">Текст *

• right – изображение выравнивается по правому краю, а текст обтекает его с левой стороны:

* <img src="foto.gif" align="right" alt="Подсказка">Текст *

• top – изображение выравнивается по верхней границе текущей строки:

* <img src="foto.gif" align="top" alt="Подсказка">Текст *

• bottom – изображение выравнивается по нижней границе текущей строки:

* <img src="foto.gif" align="bottom" alt="Подсказка">Текст *

• middle – центр изображения выравнивается по базовой линии текущей строки:

* <img src="foto.gif" align="middle" alt="Подсказка">Текст *

• hspace – отступ от изображения до текста по горизонтали:

* <img src="foto.gif" align="left" hspace="20" alt="Подсказка">Текст *

• vspace – отступ от изображения до текста по вертикали:

* <img src="foto.gif" align="left" vspace="20" alt="Подсказка">Текст *

3.19. Изображение в качестве фона

Параметр background тега <body> позволяет задать фоновый рисунок для документа:

<body background="foto.gif" bgcolor="gray">Тело документа</body>

В параметре bgcolor следует указывать цвет, близкий к цвету фонового изображения, так как резкий переход, например, от светлого тона к темному вызовет неприятное зрительные ощущения, ведь фоновое изображение может загрузиться с некоторой задержкой.

3.20. Гиперссылки

Гиперссылки позволяют нажатием кнопки мыши быстро перемещаться от одного документа к другому. Именно гиперссылки связывают

все веб-страницы в единую сеть.

3.21. Внешние гиперссылки

Внешние гиперссылки вставляются в HTML-документ с помощью тега <a>. Сам тег <a> должен быть расположен внутри блочного тега, например, <p>, или др.

Основным параметром тега <a> является href. Именно этот параметр задает URL-адрес веб-страницы, которая будет загружена при щелчке мыши на указателе. В качестве указателя может использоваться текст:

Текст ссылки

или изображение:

**
<img src="http://www.mysite.ru/foto.gif" alt="Подсказка">

Кроме HTML-документов можно ссылаться и на файлы других типов, например, изображения, архивы и т.д. При переходе по такой ссылке веб-браузер в зависимости от типа файла либо отобразит его, либо предложит сохранить. URL-адреса бывают абсолютными и относительными.

3.22. Абсолютный URL-адрес

Абсолютный URL-адрес содержит обозначение протокола, доменный или IP-адрес компьютера, путь к файлу, а также имя файла. Например:

http://www.mysite.ru/folder/file.html

Если файл находится в корневой папке, то путь может отсутствовать:

http://www.mysite.ru/file.html

Имя файла также может отсутствовать. В этом случае загружается веб-страница, заданная по умолчанию в настройках веб-сервера:

http://www.mysite.ru/ или *http://www.mysite.ru/folder/*

3.23. Относительный URL-адрес

При относительном задании URL-адреса путь определяется с учетом местоположения веб-страницы, на которой находится ссылка.

Возможны следующие варианты:

• если нужная веб-страница находится в той же папке, что и веб-страница, содержащая ссылку, то URL-адрес может содержать только имя файла. Если с веб-страницы, находящейся по адресу http://www.mysite.ru/folder1/folder2/file1.html, нужно перейти на http://www.mysite.ru/folder1/folder2/file2.html, то ссылка будет такой:

Текст ссылки

• если с веб-страницы, находящейся по адресу http://www.mysite.ru/folder1/folder2/file1.html, нужно перейти на http://www.mysite.ru/folder1/folder2/folder3/file2.html, то ссылку можно указать так:

Текст ссылки

• если с веб-страницы, находящейся по адресу http://www.mysite.ru/folder1/folder2/file1.html, нужно перейти на http://www.mysite.ru/folder1/file2.html, то ссылка будет такой:

Текст ссылки

А при переходе с http://www.mysite.ru/folder1/folder2/folder3/file1.html на http://www.mysite.ru/folder1/file2.html – такой:

Текст ссылки

Очень часто необходимо загрузить документ в новое окно веб-браузера. Для этого в параметре target тега <a> следует указать значение _blank:

Ссылка

3.24. Внутренние гиперссылки

С помощью внутренних гиперссылок можно создать ссылки на разные разделы текущей веб-страницы. Если документ очень большой, то наличие внутренних гиперссылок позволяет быстро перемещаться между его разделами. Внутренняя гиперссылка также вставляется при помощи тега <a> с одним отличием – параметр href содержит имя указателя, а не URL-адрес. Перед именем указателя ставится знак #:

Глава 1

Указатель создается с помощью тега <a>, но вместо параметра href

используется параметр name, который задает имя указателя:

```
<a name="chapter1"></a>
```

Иногда указатель называют "якорем". Также можно сослаться на "якорь" другого документа:

```
<a href="http://www.mysite.ru/file.html#chapter6">Текст</a>
```

Структура документа с внутренними ссылками приведена ниже.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Создание внутренних ссылок</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<h1>Название документа</h1>
<h2>Оглавление</h2>
<ul>
<li><a href="#chapter1">Глава 1</a></li>
<li><a href="#chapter2">Глава 2</a></li>
<li><a href="#chapter3">Глава 3</a></li>
<li><a href="#chapter4">Глава 4</a></li>
</ul>
<h2><a name="chapter1"></a>Глава 1</h2>
<p>Содержание главы 1</p>
<h2><a name="chapter2"></a>Глава 2</h2>
<p>Содержание главы 2</p>
<h2><a name="chapter3"></a>Глава 3</h2>
<p>Содержание главы 3</p>
<h2><a name="chapter4"></a>Глава 4</h2>
<p>Содержание главы 4</p>
</body>
</html>
```

3.25. Гиперссылки на адрес электронной почты

Ссылка на адрес электронной почты выглядит так:

Текст

Вместо URL-адреса указывается адрес электронной почты, перед которым добавляется слово "mailto:".

3.26. Таблицы

В HTML-документе таблицы используются как средство представления данных или как элемент оформления страницы, с помощью которого можно точно разместить на странице текст и графику.

Таблица вставляется в HTML-документ с помощью парного тега <table>.

Отдельная ячейка таблицы описывается тегом <td>, а ряд ячеек – с помощью тега <tr>. Тег <caption> позволяет задать заголовок таблицы.

Приведем пример структуры, описывающей таблицу с заголовком, значениями в ячейках, выровненными по центру, ячейками таблицы, пронумерованными от 1 до 4.

```
<table border="1" width="200">
<caption>Заголовок таблицы</caption>
<tr>
<td align="center">1</td>
<td align="center">2</td>
</tr>
<tr>
<td align="center">3</td>
<td align="center">4</td>
</tr>
</table>
```

Тег <table> имеет следующие параметры:

• border управляет отображением линий сетки таблицы, а также задает толщину рамки вокруг таблицы. По умолчанию сетка не отображается:

```
<table><!-- Здесь сетка не отображается -->
<table border="0"><!-- Здесь сетка не отображается -->
```

<table border="5"><!-- В этом случае сетка отображается, а толщина рамки вокруг таблицы равна 5 пикселам -->

• cellspacing задает толщину линий сетки внутри таблицы, точнее сказать, расстояние между рамками соседних ячеек. По умолчанию параметр имеет значение 2. Если параметру присвоить значение 0, то рамки смежных ячеек сольются в одну линию:

<table cellspacing="0">

• cellpadding указывает размер отступа между рамкой ячейки и данными внутри ячейки. По умолчанию параметр имеет значение 1:

<table cellpadding="2">

• width определяет ширину таблицы в пикселах или в процентах от размера окна:

<table width="200">

<table width="100%">

• bgcolor указывает цвет фона таблицы:

<table bgcolor="silver">

<table bgcolor="#C0C0C0">

3.27. Заголовок таблицы

Тег <caption> позволяет задать заголовок таблицы. Он имеет единственный параметр align, который может принимать одно из двух значений:

• top – заголовок помещается над таблицей:

<caption align="top">Заголовок таблицы</caption>

• bottom – заголовок располагается под таблицей:

<caption align="bottom">Заголовок таблицы</caption>

3.28. Строки таблицы

С помощью парного тега <tr> описываются строки таблицы. Он имеет следующие параметры:

• align указывает горизонтальное выравнивание текста в ячейках таблицы.

Параметр может принимать следующие значения:

• left – по левому краю (по умолчанию): *<tr align="left">*

• right – по правому краю: *<tr align="right">*

• center – по центру: *<tr align="center">*

• justify – по ширине: *<tr align="justify">*

• valign определяет вертикальное выравнивание текста в ячейках таблицы. Он может принимать следующие значения:

– top – по верхнему краю: *<tr valign="top">*

– middle – по центру: *<tr valign="middle">*

– bottom – по нижнему краю: *<tr valign="bottom">*

– baseline – по базовой линии: *<tr valign="baseline">*

3.29. Ячейки таблицы

С помощью тега <td> описываются ячейки таблицы. Тег <td> имеет следующие параметры:

• align и valign определяют горизонтальное и вертикальное выравнивание текста в ячейках таблицы;

• width и height определяют ширину и высоту ячейки в пикселах или в процентах;

• bgcolor указывает цвет фона ячейки;

• colspan задает количество объединяемых ячеек по горизонтали;

• rowspan указывает количество объединяемых ячеек по вертикали.

В качестве примера объединения ячеек возьмем таблицу, описанную выше, и объединим горизонтально расположенные ячейки 1 и 2 в одну.

```
<table border="1" width="200">
<caption>Заголовок таблицы</caption>
<tr>
<td align="center" colspan="2">1 и 2 объединены</td>
</tr>
<tr>
<td align="center">3</td>
```

<td align="center">4</td>
</tr>
</table>

Таким образом, строка <td align="center">1</td> была заменена строкой <td align="center" colspan="2">1 и 2 объединены</td>, при этом строка <td align="center">2</td> была удалена.

Теперь объединим вертикально расположенные ячейки 1 и 3 в одну.

<table border="1" width="200">
<caption>Заголовок таблицы</caption>
<tr>
<td align="center" rowspan="2">1 и 3 объединены</td>
<td align="center">2</td>
</tr>
<tr>
<td align="center">4</td>
</tr>
</table>

В этом примере строка <td align="center">1</td> была заменена строкой <td align="center" rowspan="2">1 и 3 объединены</td>, при этом строка <td align="center">3</td> была удалена.

Тег <th> позволяет указать ячейки, которые являются заголовочными. Содержимое таких ячеек выделяется полужирным шрифтом и размещается по центру. Во всем остальном тег <th> аналогичен тегу <td>.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Выделение ячеек таблицы</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<table border="1" width="500">

<caption>Заголовок таблицы</caption>
<tr>
<th>Марка</th>
<th>Цвет</th>
<th>Год выпуска</th>
</tr>
<tr>
<td>ВАЗ-2109</td>
<td>Красный</td>
<td>2008</td>
</tr>
<tr>
<td>Москвич-412</td>
<td>Белый</td>
<td>1978</td>
</tr>
</table>
</body>
</html>

Практические задания

1. Придумать название и содержание своего собственного небольшого сайта (собственная домашняя страница, сайт какой-либо организации, небольшой интернет-магазин и т.д.).

2. Создать первую пустую веб-страницу данного сайта. Она должна быть корректной HTML страницей, содержащей теги html, head, title, body. Тег title должен содержать придуманное название сайта.

3. Придумать содержание данной страницы. Содержание должно включать в себя некоторое приветственное слово посетителям, текст-описание данного ресурса, e-mail адрес разработчика сайта, небольшое меню сайта, ведущее на страницы разделов сайта (два-три раздела; страницы самих разделов сайта необходимо наполнить информацией по соответствующей тематике). В тексте необходимо использовать теги h1, b, i,

center и др.

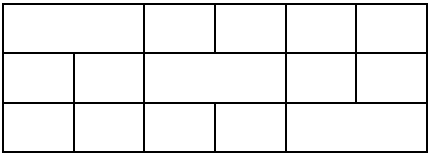
4. На вновь созданной странице разместить содержимое, включающее в себя: комбинированный список (т.е. содержащий как нумерованные, так и ненумерованные подсписки), содержащий 3 уровня вложенности и не менее 10-15 пунктов (суммарно на всех уровнях вложенности). Списки должны отличаться по типу – нумерованные, ненумерованные и нумерованные/ненумерованные (по выбору).

5. На вновь созданной странице разместить содержимое, включающее в себя список и таблицу согласно вашему варианту и различным образом отформатированный текст (использовать теги <p>,
, <hn>, <b>, <i>, <u>,<address>, <blockquote>,, <sub>, <sup>, <font>); − графическое изображение (<img>); − гиперссылки: на другой html-документ, в пределах создаваемого html-документа, на e-mail (<a>); − бегущую строку (<marquee>).

Варианты заданий

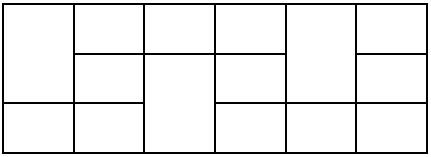
Вариант 1

3. Зима
4. Весна
5. Лето
6. Осень



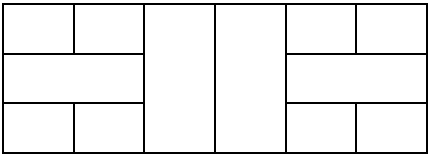
Вариант 2

- Зима
- Весна
- Лето
- Осень



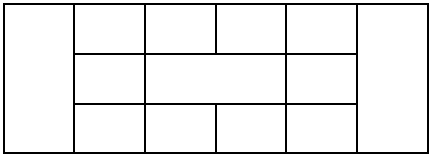
Вариант 3

c. Зима
ci. Весна
cii. Лето
ciii. Осень



Вариант 4

y. Зима
z. Весна
aa. Лето
ab. Осень



Контрольные вопросы

1. Что такое HTML-документ?
2. Структура HTML-документа?
3. Что называется меткой (тэгом)?
4. Что такое атрибуты?
5. Назначение метки <html>...</html>.
6. Назначение метки <head>...</head>.
7. Назначение метки <title>...</title>.
8. Назначение метки <body>...</body>.
9. Назначение меток <h1>...</h1> – <h6>...</h6>.
10. Назначение метки ... .
11. Назначение атрибута align. Какие значения он может принимать?
12. Назначение метки
?
13. Назначение метки <hr>?
14. Назначение метки <!--...-->?
15. Назначение метки .... Какие значения она может принимать? Каковы ее атрибуты?
16. Как организовать маркированный список?
17. Как организовать нумерованный список?
18. Как организовать списки определений?
19. Что такое вложенные списки?
20. Как организовать форматированный текст с отступами?
21. Что такое гипертекст?
22. Как представить изображение на веб-странице?
23. Какие существуют атрибуты при представлении изображения?

24. Что такое анкер?

25. Как установить цвет фона документа?

26. Как установить цвет текста документа?

27. Как определить цвет выделенного элемента текста, при нажатии на который происходит переход по гипертекстовой ссылке?

28. Как определить цвет ссылки на документ, который уже был просмотрен ранее?

29. Как определить цвет ссылки в момент, когда на нее указывает курсор мыши и нажата ее правая кнопка, то есть непосредственно перед переходом по ссылке?

30. Как кодируется цвет?

31. Заголовок HTML-документа.

32. Перечислите наиболее известные META-инструкции?

33. Атрибуты метки <table>.

34. Атрибуты метки caption>.

35. Атрибуты метки <tr>.

36. Атрибуты метки <td>.

4. СОЗДАНИЕ ФОРМЫ И НАВИГАЦИОННОЙ КАРТЫ

4.1. Карты-изображения

С помощью карт-изображений можно создать уникальную панель навигации. В качестве ссылок на разделы сайта будут использоваться области на обычном изображении формата GIF или JPG.

Однако такой подход не лишен некоторых недостатков, к которым можно отнести:

• веб-браузеры не выделяют другим цветом посещенные ссылки;

• поскольку вместо текстовых ссылок используются изображения, увеличится время загрузки страницы;

• если пользователь отключит графику, он не увидит панель навигации.

Код для вставки карты-изображения состоит из двух частей.

Первая часть с помощью тега <img> вставляет изображение в веб-страницу. Параметр usemap указывает, что изображение является картой. В качестве значения параметра указывается URL-адрес. Если описание карты расположено в том же HTML-документе, то указывается название раздела конфигурации карты, перед которым добавляется символ "#".

Вторая часть, являющаяся описанием конфигурации карты, расположена между тегами <map> и </map>. Активная область карты описывается с помощью тега <area>.

Парный тег <map> служит для описания конфигурации областей карты изображения. У тега <map> есть единственный параметр – name. Значение параметра name должно соответствовать имени в параметре usemap тега <img>.

Одинарный тег <area> описывает одну активную область на карте. Если одна активная область перекрывает другую, то будет реализована первая ссылка из списка областей. Тег имеет следующие параметры:

1)shape задает форму активной области и может принимать 4 значения:

• rect – активная область в форме прямоугольника (значение по умолчанию): *<area shape="rect" alt="Подсказка">*;

• circle – активная область в форме круга: *<area shape="circle"*

alt="Подсказка">;

• poly – активная область в форме многоугольника: *<area shape="poly" alt="Подсказка">*;

• default – активная область занимает всю площадь изображения. Данное значение не поддерживается браузером Internet Explorer: *<area shape="default" alt="Подсказка">*.

2) coords определяет координаты точек отдельной активной области. Координаты указываются относительно верхнего левого угла изображения и задаются следующим образом:

• для области типа "rect" задаются координаты верхнего левого и правого нижнего углов прямоугольника (координаты указываются через запятую): *<area shape="rect" coords="x1, y1, x2, y2" alt="Подсказка">*. Здесь x1 и y1 – координаты левого верхнего угла, а x2 и y2 – координаты правого нижнего угла. Например:
<area shape="rect" coords="0, 0, 120, 120" alt="Подсказка">;

• для области типа "circle" указываются координаты центра круга и радиус: *<area shape="circle" coords="x1, y1, r1" alt="Подсказка">*. Здесь x1 и y1 – координаты центра круга, а r1 – радиус круга. Например:
<area shape="circle" coords="60, 60, 30" alt="Подсказка">;

• для области типа "poly" перечисляются координаты вершин многоугольника: *<area shape="poly" coords="x1, y1, x2, y2, x3, y3" alt="Подсказка">*. Здесь x1, y1, x2, y2, x3, y3 – координаты вершин многоугольника (в данном случае треугольника). Можно задавать и большее количество вершин, т.е. можно описать активную область практически любой формы. Координаты последней вершины не обязательно должны совпадать с первой. Например:
<area shape="poly" coords="10, 100, 60, 10, 100, 100" alt="Подсказка">.

3) href указывает URL-адрес ссылки. Может быть указан абсолютный или относительный адрес ссылки. Например:
<area shape="circle" coords="60, 60, 30" alt="Подсказка"
href="http://www.mysite.ru/chapter1.html">
<area shape="circle" coords="60, 60, 30" alt="Подсказка" href="chapter1.html">

4)nohref указывает, что активная область не имеет ссылки. Например:

```
<area shape="circle" coords="60, 120, 60" alt="Подсказка" nohref>
<area shape="rect" coords="0, 0, 240, 240" alt="Подсказка"
href="http://www.mysite.ru/chapter1.html">
```

В этом примере активной областью является вся площадь изображения 120×240 за исключением круга радиусом 60 в центре изображения.

5)target указывает, куда будет загружен документ при переходе по ссылке. Может быть указано имя фрейма или одно из зарезервированных значений – _blank, _top, _self или _parent. Например:

```
<area shape="rect" coords="0, 0, 240, 240" href="chapter1.html" target=
"_blank" alt="Подсказка">
```

Пример использования карт-изображений.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Использование карт-изображений</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<h1 align="center">Название документа</h1>
<img src="foto.gif" alt="Подсказка" width="120" height="240" border="1"
usemap="#karta">
<map name="karta">
<area shape="rect" coords="0,0,120,120" href="chapter1.html"
target="chapter" alt="Верхняя часть рисунка">
<area shape="rect" coords="0,120,240,240" href="chapter2.html"
target="chapter" alt="Нижняя часть рисунка">
<area shape="default" alt="" nohref>
</map>
</body>
</html>
```

4.2. Формы

Формы используются для пересылки данных от пользователя к веб-серверу. Для создания форм может быть использован язык HTML.

Пример создания формы регистрации сайта в каталоге ресурсов сети Интернет представлен ниже.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Пример использования форм</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<h1>Пример формы регистрации сайта</h1>
<form action="file.php" method="POST" enctype="multipart/form-data">
<div>
Логин:<br>
<input type="text" name="pole1"><br> Пароль:<br>
<input type="password" name="pole2" value="Пароль"><br> URL-адрес
сайта:<br>
<input type="text" name="pole3" value="http://" size="20"><br>
Название сайта:<br>
<input type="text" name="pole4" size="20"><br> Описание сайта:<br>
<textarea name="pole5" rows="10" cols="15"></textarea><br>
Тема сайта:<br>
<select name="pole6">
<option value="0" selected>Тема не выбрана</option>
<option value="1">Тема1</option>
<option value="2">Тема2</option>
<option value="3">Тема3</option>
</select><br>
Баннер 88*31:<br>
```

*<input type="file" name="pole7" size="20">

*
<input type="reset" value="Очистить">
<input type="submit" value="Отправить">
* *
</form>
</body>
</html>

Созданный html-файл следует сохранить под именем, например, forma.html. При последующем открытии этого файла в окне веб-браузера будет отображена форма, состоящая из пяти текстовых полей (Логин, Пароль, URL-адрес сайта, Название сайта и Описание сайта); списка значений (Тема сайта); поля выбора файла с кнопкой Обзор (под надписью "Баннер 88*31:"); двух кнопок Очистить и Отправить.

Кнопка Очистить возвращает все значения формы к первоначальным. Кнопка Отправить позволяет отправить данные, введенные пользователем, программе URL-адрес которой указан в параметре action тега <form>, в данном случае file.php. Эта программа расположена на веб-сервере. Программа обработает данные и либо добавит сайт в каталог и выдаст подтверждение об успешной регистрации, либо выдаст сообщение об ошибке, если обязательное поле не заполнено или заполнено неверно. В нашем случае программы обработки нет, и отправка данных формы ни к чему не приведет, точнее, приведет к ошибке "Файл не найден".

4.3. Структура документа с формами

Форма добавляется в HTML-документ при помощи парного тега <form>. Внутри тегов <form> и </form> могут располагаться теги <input>, <textarea> и <select>, вставляющие в форму элементы управления. Например:

<form action="file.php">
<input type="text">
<textarea></textarea>
<select>
<option></option>

</select>
<input type="file">
<input type="reset">
<input type="submit">
</form>

Тег <form> имеет следующие параметры:

1)action задает URL-адрес программы обработки формы. Может задаваться в абсолютной или относительной форме. Например:

<form action="file.php">
<form action="http://www.mysite.ru/file.php">

2)method определяет, как будут пересылаться данные от формы к вебсервера. Может принимать два значения – GET и POST:

– GET – данные формы пересылаются путем их добавления к URL-адресу после знака "?" в формате [Имя параметра]=[Значение параметра]. Пары параметр=значение отделяются друг от друга символом амперсанда (&). Например:

http://www.mysite.ru/file.php?pole1=Login&pole2=Password

Все специальные символы, а также буквы, отличные от латинских (например, буквы русского языка), кодируются в формате %nn, а пробел заменяется знаком "+". Например, фраза "каталог сайтов" будет выглядеть следующим образом:

%EA%E0%F2%E0%EB%EE%E3+%F1%E0%E9%F2%EE%E2

А если эта фраза является значением поля с именем pole1, то строка запроса будет такой:

http://www.mysite.ru/file.php?pole1=%EA%E0%F2%E0%EB%EE%E3+%F1%E0%E9%F2%EE%E2&pole2=Password

В теге <form> значение GET для параметра method задается так:

<form action="http://www.mysite.ru/file.php" method="GET">

Метод GET применяется, когда объем пересылаемых данных невелик, так как существует предел длины URL-адреса. Длина не может превышать 256 символов;

– POST предназначен для пересылки данных большого объема,

файлов и конфиденциальной информации (например, паролей). Задается так:
<form action="http://www.mysite.ru/file.php" method="POST">

3)enctype задает MIME-тип передаваемых данных. Может принимать два значения:

– application/x-www-form-urlencoded – применяется по умолчанию. Например:
<form action="http://www.mysite.ru/file.php" method="POST"
enctype="application/x-www-form-urlencoded">

– multipart/form-data – указывается при пересылке файлов веб-серверу. Например:
<form action="http://www.mysite.ru/file.php" method="POST"
enctype="multipart/form-data">

4)name задает имя формы. Например:
<form action="file.php" name="form1">

5)target указывает, куда будет помещен документ, являющийся результатом обработки данных формы веб-сервером. Параметр может содержать имя фрейма или одно из зарезервированных значений – _blank, _top, _self или _parent. Например:
<form action="http://www.mysite.ru/file.php" method="POST"
enctype="multipart/form-data" target="_blank">

Тег <input> позволяет вставить в форму элементы управления, например, текстовое поле, кнопку или флажок. Этот тег имеет параметры:

1)type задает тип элемента. В зависимости от значения этого поля создаются следующие элементы формы:

- text – текстовое поле ввода. Например: *<input type="text">*
- password – текстовое поле для ввода пароля, в котором все вводимые символы заменяются звездочкой. Например: *<input type="password">*
- file – поле ввода имени файла с кнопкой Обзор. Позволяет отправить файл на веб-сервер. Например: *<input type="file">*
- checkbox – поле для установки флажка, который можно установить или сбросить. Например: *<input type="checkbox">*
- radio – элемент-переключатель (иногда их называют радиок-

нопками). Например: *<input type="radio">*

• reset – кнопка, при нажатии на которую вся форма очищается, т.е. все элементы формы получают значения по умолчанию. Например: *<input type="reset">*

• submit – кнопка, при нажатии на которую происходит отправка данных, введенных в форму. Например: *<input type="submit">*

• button – обычная командная кнопка: *<input type="button">* Такую кнопку можно использовать только с прикрепленным к ней скриптом.

• hidden – скрытый элемент, значение которого отправляется вместе со всеми данными формы. Элемент скрыт от пользователя, но позволяет сохранить, например, данные из предыдущей формы, если пользователь последовательно заполняет несколько форм, или уникальное имя пользователя: *<input type="hidden">*

2)name задает имя элемента управления. Оно должно обязательно указываться латинскими буквами (например, pole) или комбинацией латинских букв и цифр (например, pole1). Имя элемента не должно начинаться с цифры. Например: *<input type="text" name="pole1">*

3)disabled запрещает доступ к элементу формы. При наличии параметра элемент отображается серым цветом. Например: *<input type="text" name="pole1" disabled>*.

Остальные параметры специфичны для каждого отдельного элемента.

Для текстового поля и поля ввода пароля применяются параметры:

• value задает текст поля по умолчанию. Например:
<input type="text" name="pole1" value="http://">

• maxlength указывает максимальное количество символов, которое может быть введено в поле. Например:
<input type="text" name="pole1" value="http://" maxlength="100">

• size определяет видимый размер поля ввода. Например:
<input type="text" name="pole1" value="http://" maxlength="100" size="20">

• readonly указывает, что поле доступно только для чтения. При наличии параметра значение поля изменить нельзя. Например:
<input type="text" name="pole1" readonly>

Для кнопок используется один параметр:

• value задает текст, отображаемый на кнопке. Например:

<input type="submit" value="Отправить">

Для скрытого поля указывается один параметр:

• value определяет значение скрытого поля. Например:

<input type="hidden" name="pole1" value="1">

Для полей-флажков используются следующие параметры:

• value задает значение, которое будет передано веб-серверу, если флажок отмечен. Если флажок снят, значение не передается. Если параметр не задан, используется значение по умолчанию – on. Например:

<input type="checkbox" name="check1" value="yes">Текст

• checked указывает, что флажок по умолчанию отмечен. Например:

<input type="checkbox" name="check1" value="yes" checked>Текст

Элементы checkbox можно объединить в группу. Для этого необходимо установить одинаковое значение параметра name. Например:

<input type="checkbox" name="check[]" value="1">Текст 1

<input type="checkbox" name="check[]" value="2">Текст 2

<input type="checkbox" name="check[]" value="3">Текст 3

При описании элемента-переключателя используются такие параметры:

• value указывает значение, которое будет передано веб-серверу, если переключатель выбран. Если ни одно из значений не выбрано, никаких данных передано не будет. Например:

<input type="radio" name="radio1" value="yes">Текст

• checked обозначает переключатель, выбранный по умолчанию. Например: *<input type="radio" name="radio1" value="yes" checked>Текст*

Элемент-переключатель может существовать только в составе группы подобных элементов, из которых может быть выбран только один. Для объединения переключателей в группу необходимо установить одинаковое значение параметра name и разное значение параметра value. Например:

*Укажите ваш пол:
*

<input type="radio" name="radio1" value="male" checked>Мужской

<input type="radio" name="radio1" value="female">Женский

Парный тег <textarea> создает внутри формы поле для ввода многострочного текста. В окне веб-браузера поле отображается в виде прямоугольной области с полосами прокрутки. Тег имеет следующие параметры:

• name – уникальное имя поля. Например:

<textarea name="pole2"> Текст по умолчанию </textarea>

• cols – число столбцов видимого текста. Например:

<textarea name="pole2" cols="15"> Текст по умолчанию </textarea>

• rows – число строк видимого текста. Например:

<textarea name="pole2" cols="15" rows="10"> Текст по умолчанию </textarea>

Тег <select> создает внутри формы список с возможными значениями. Например:

```
<select>
<option>Элемент1</option>
<option>Элемент2</option>
</select>
```

Тег <select> имеет следующие параметры:

• name задает уникальное имя списка. Например: *<select name="select1">*

• size определяет число одновременно видимых элементов списка. Например: *<select name="select1" size="3">*

По умолчанию параметр size имеет значение 1.

• multiple указывает, что из списка можно выбрать несколько элементов одновременно. Например: *<select name="select[]" size="3" multiple>*

Внутри тегов <select> и </select> располагаются теги <option>, с помощью которых описывается каждый элемент списка.

Тег <option> имеет следующие параметры:

• value задает значение, которое будет передано веб-серверу, если пункт списка выбран. Если параметр не задан, отправляется текст этого пункта. Например:

```
<select name="select1">
<option value="val1">Элемент1</option>
<option>Элемент2</option>
</select>
```

Если выбран пункт "Элемент1", то отправляется select1="val1".

Если выбран пункт "Элемент2", то отправляется select1="Элемент2".

• selected указывает, какой пункт списка выбран изначально. Например:

```
<select name="select1">
<option value="val1">Элемент1</option>
<option selected>Элемент2</option>
</select>
```

С помощью тега <optgroup> можно объединить несколько пунктов в группу. Название группы указывается в параметре label. Например:

```
<select name="select1">
<optgroup label="Отечественные">
<option value="1">ВАЗ</option>
<option value="2">ГАЗ</option>
</optgroup>
<optgroup label="Зарубежные">
<option value="3">BMW</option>
<option value="4">Audi</option>
</optgroup>
</select>
```

С помощью тега <label> можно указать пояснительную надпись для элемента формы. Тег имеет два параметра:

• for позволяет указать идентификатор элемента, к которому привязана надпись. Точно такой же идентификатор должен быть указан в параметре id элемента формы. Например:

```
<label for="pass1">Пароль*:</label>
<input type="password" name="pass1" id="pass1">
```

Параметр id могут иметь практически все теги. В большинстве случаев он предназначен для доступа к элементу из скрипта.

Если элемент формы разместить внутри тега <label>, то надпись будет связана с элементом и без указания параметра for. Например:

```
<label>Пароль*:
<input type="password" name="pass1" id="pass1"> </label>
```

Accesskey задает клавишу быстрого доступа. При нажатии этой клавиши одновременно с клавишей <Alt> элемент окажется в фокусе ввода. Например:

```
<label accesskey="N">Пароль*:
<input type="password" name="pass1" id="pass1">
</label>
```

Браузеры Opera и Firefox не поддерживают параметр accesskey.

4.4. Группировка элементов формы

Парный тег <fieldset> позволяет сгруппировать элементы формы. Вокруг группы веб-браузеры отображают рамку. На линии рамки с помощью тега <legend> можно разместить надпись. Например:

```
<fieldset>
<legend>Пол</legend>
Мужской <input type="radio" name="sex" value="1">
Женский <input type="radio" name="sex" value="2">
</fieldset>
```

Практические задания

1. Используя произвольное изображение, создайте навигационную карту.
2. Создайте регистрационную форму, включающую следующие элементы: текстовые поля, поля для ввода пароля, поля множественного выбора, радиокнопки, текстовое поле (TEXTAREA), кнопки для очистки элементов формы и отправки данных на сервер.

Контрольные вопросы

1. Что такое навигационная карта?
2. Каково функциональное назначение элементов MAP и AREA?
3. Какие значения может принимать атрибут shape?
4. Какие значения может принимать атрибут type элемента input?
5. Для чего необходим элемент label?

5. ИСПОЛЬЗОВАНИЕ КАСКАДНЫХ ТАБЛИЦ СТИЛЕЙ ДЛЯ ОФОРМЛЕНИЯ ТЕКСТОВОГО ФРАГМЕНТА (ФОРМАТИРОВАНИЯ ТЕКСТА)

5.1. Основные понятия

Каскадные таблицы стилей (CSS – Cascading Style Sheets) позволяют существенно расширить возможности языка HTML за счет более гибкого управления форматированием веб-страницы.

В качестве примера рассмотрим параметр size тега :

Текст

В языке HTML размер шрифта указывается в условных единицах от 1 до 7. По умолчанию веб-браузером устанавливается размер шрифта, равный 3 условным единицам. Однако размер шрифта в пунктах, используемый по умолчанию, отличается в разных веб-браузерах. Следовательно, отображение веб-страницы также будет отличаться в разных веб-браузерах.

С помощью параметра style CSS можно точно задать размер шрифта в пунктах. Например: *Текст*

Применение стилей позволяет задавать точные характеристики практически всех элементов веб-страницы, таким образом можно контролировать внешний вид веб-страницы в окне веб-браузера.

Значение параметра style (font-size: 12pt) называется определением стиля или стилем. Элемент определения стиля (font-size) называется атрибутом. Каждый атрибут имеет значение (12pt), указываемое после двоеточия.

Совокупность определений стилей, вынесенных в заголовок HTML-документа или в отдельный файл, называют таблицей стилей.

5.2. Способы встраивания стиля

Задать стиль можно тремя способами: встроить определение стиля в тег, встроить определение стилей в заголовок HTML-документа или вынести таблицу стилей в отдельный файл.

Встраивание стиля в тег

Определение стиля встраивается в любой тег с помощью параметра style,

который имеют все теги. Например: *Текст*

Если определение стиля состоит из нескольких атрибутов, то они указываются через точку с запятой. Например:

Текст

Если какое-либо значение атрибута требует наличия кавычек, то оно указывается в апострофах. Например:

<font style="font-size: 12pt; color: red;
font-family: 'Times New Roman'">Текст

Встраивание стилей в заголовок HTML-документа

Все определения стилей можно сгруппировать. В этом случае стили указываются между тегами <style> и </style>. Сам тег <style> должен быть расположен в разделе HEAD HTML-документа. Например:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Пример использования стилей</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<style type="text/css">
.text1 { font-size: 12pt; color: red; font-family: "Arial" }
font { font-size: 12pt; color: green; font-family: "Arial" }
font.text2 { font-size: 12pt; color: blue; font-family: "Arial" }
</style>
</head>
<body>
<font class="text1">Текст1</font><br><!-- Красный текст -->
<font>Текст2</font><br><!-- Зеленый текст -->
<font class="text2">Текст3</font><br><!-- Синий текст -->
<p class="text2">Текст4</p><!-- Цвет по умолчанию -->
<p class="text1">Текст5</p><!-- Красный текст -->
</body>
</html>
```

Атрибуты определения стиля, указанные между тегами <style> и </style>, заключаются в фигурные скобки. Если атрибутов несколько, то они перечисляются через точку с запятой:

<Селектор> { <Атрибут 1>: <Значение 1>; ..., <Атрибут N>: <Значение N> }

В параметре <Селектор> могут быть указаны следующие селекторы:

1. * – все теги. Например, чтобы убрать все внешние и внутренние отступы следует написать: ** { margin: 0; padding: 0 }*

2. Тег – все теги, имеющие указанное имя. Например:

```
font { font-size: 12pt; color: green; font-family: "Arial" } ...
<font>Текст2</font><!-- Зеленый текст -->
```

3. Класс – все теги, имеющие указанный класс. Например:

```
.text1 { font-size: 12pt; color: red; font-family: "Arial" } ...
<font class="text1">Текст1</font><!-- Красный текст -->
<p class="text1">Текст2</p><!-- Красный текст -->
```

И "Текст1" и "Текст2" будут красного цвета, хотя они находятся в разных тегах;

4. Тег.Класс – все теги, имеющие указанный класс. Например:

```
font.text2 { font-size: 12pt; color: blue } ...
<font class="text2">Текст1</font><!-- Синий текст -->
```

Если имя класса указать в другом теге, то он не будет определен. Например: * Текст2 *

В данном случае фрагмент текста "Текст2" не будет отображен синим цветом, так как имя класса text2 применяется только к тегу <font>;

5. #Идентификатор – тег с указанным идентификатором. Например:

```
#txt1 { color: red } ...
<p id="txt1">Текст</p>
```

6. Тег#Идентификатор – идентификатор, указанный в определенном теге. Если идентификатор находится в другом теге, то элемент будет проигнорирован. Например:

```
p#txt1 { color: red } ...
<p id="txt1">Текст</p>
```

Стилевой класс можно привязать сразу к нескольким тегам. В этом случае селекторы указываются через запятую. Например:

h1, h2 { font-family: "Arial" }

Привязаться к другим элементам можно следующими способами:

• Селектор1 Селектор2 – все элементы, соответствующие параметру Селектор2, которые располагаются внутри контейнера, соответствующего параметру Селектор1. Например: *div a { color: red }*

Цвет текста ссылки станет красным, если тег <a> находится внутри тега . Например: * Ссылка *

• Селектор1 > Селектор2 – все элементы, соответствующие параметру Селектор2, которые являются дочерними для контейнера, соответствующего параметру Селектор1. Например: *div > a { color: red }*

Цвет текста ссылки станет красным, если тег <a> находится внутри тега и не вложен в другой тег. Например:

* *
*Ссылка 1
*
*Ссылка 2 *

В этом примере только первая ссылка станет красного цвета, так как вторая ссылка расположена внутри тега ;

• Селектор1 + Селектор2 – элемент, соответствующий параметру Селектор2, который является соседним для контейнера, соответствующего параметру Селектор1, и следует сразу после него. Например: *div+a { color: red }*

Цвет текста ссылки станет красным, если тег <a> следует сразу после элемента div. Например: * Текст Ссылка *

При необходимости можно составлять выражения из нескольких селекторов. Например:

div span a { color: red }

Цвет текста ссылки станет красным, если тег <a> расположен внутри тега , а тот в свою очередь вложен в тег . Например:

* *
*Ссылка 1
*
**

*Ссылка 2
*
* *

В этом примере только ссылка 2 будет красного цвета.

Для привязки к параметрам тегов применяются следующие селекторы:

• [Параметр] – элементы с указанным параметром. Например: *a[id] { color: red }*

Цвет текста ссылки станет красным, если тег <a> имеет параметр id. Например: *Ссылка 1*

• [Параметр="Значение"] – элементы, у которых параметр точно равен значению. Например: *a[href="link1.html"] { color: red }*

Цвет текста ссылки станет красным, если параметр href тега <a> имеет значение "link1.html";

• [Параметр^='Значение'] – элементы, у которых параметр начинается с указанного значения. Например: *a[href^="li"] { color: red }*

Цвет текста ссылки станет красным, если значение параметра href тега <a> начинается с "li";

• [Параметр$='Значение'] – элементы, у которых параметр заканчивается указанным значением. Например: *a[href$=".html"] { color: red }*

Цвет текста ссылки станет красным, если значение параметра href тега <a> имеет расширение ".html";

• [Параметр*='Значение'] – элементы, у которых параметр содержит указанный фрагмент значения. Например: *a[href*="link"] { color: red }*

Цвет текста ссылки станет красным, если значение параметра href тега <a> содержит фрагмент "link".

В качестве селектора могут быть также указаны следующие псевдоэлементы:

• first-letter – задает стиль для первой буквы. Например:
p:first-letter { font-size: 150%; font-weight: bold; color: red }

• first-line – задает стиль для первой строки. Например:
p:first-line { font-weight: bold; color: red }

• before и :after – позволяют добавить текст в начало и конец элемента соответственно. Добавляемый текст должен быть указан в атрибуте

content. Например:

p:before { content: "before "; }

p:after { content: " after"; } ...

* Текст *

Результат: *before Текст after*

Вынесение таблицы стилей в отдельный файл

Таблицу стилей можно вынести в отдельный файл. Файл с таблицей стилей обычно имеет расширение *.css и может редактироваться любым текстовым редактором. Задать расширение файлу можно точно так же, как и при создании файла с расширением html.

Ниже представлен пример подключения таблицы стилей, вынесенной в отдельный файл style.css, к html-документу test.html.

/ Так можно вставить комментарий */*

.text1 { / Стиль для элементов с class="text1" */*

font-size: 12pt; / Размер шрифта */*

color: red; / Цвет текста */*

font-family: Arial / Название шрифта */ }*

font { / Стиль для всех тегов FONT */*

font-size: 12pt; / Размер шрифта */*

color: green; / Цвет текста */*

font-family: Arial / Название шрифта */ }*

font.text2 { / Стиль для тегов FONT с class="text2" */*

font-size: 12pt; / Размер шрифта */*

color: blue; / Цвет текста */*

font-family: Arial / Название шрифта */ }*

Содержимое файла html:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<title>Пример использования стилей</title>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">

```
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<font class="text1">Текст1</font><br><!-- Красный текст -->
<font>Текст2</font><br><!-- Зеленый текст -->
<font class="text2">Текст3</font><br><!-- Синий текст -->
<p class="text2">Текст4</p><!-- Цвет по умолчанию -->
<p class="text1">Текст5</p><!-- Красный текст -->
</body>
</html>
```

Оба файла следует сохранить в одной папке.

Отдельный файл с таблицей стилей прикрепляется к HTML-документу с помощью одинарного тега <link>. В параметре href указывается абсолютный или относительный URL-адрес файла, а в параметре rel должно быть значение stylesheet, показывающее, что присоединяемый таким образом документ содержит таблицу стилей. Например:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

Подключить внешний CSS-файл можно также с помощью правила @import. Например:

```
@import url(<URL-адрес>)[ <Тип устройства>];
@import <URL-адрес>[ <Тип устройства>];
```

Правило @import: должно быть расположено внутри тега <style>:

```
<style type="text/css">
@import url("style.css");
</style>
```

В необязательном параметре <Тип устройства> можно указать устройство, для которого предназначена подключаемая таблица стилей. Например, all – для любых устройств, print – для предварительного просмотра и распечатки документа. Пример:

```
<style type="text/css">
@import "style.css" print;
</style>
```

Таблицу стилей, вынесенную в отдельный файл, можно использовать в нескольких HTML-документах.

5.3. Приоритет применения стилей

В CSS существуют правила, описывающие приоритет стилей:

• стиль, заданный таблицей стилей, будет отменен, если в HTML-коде явно описано форматирование блока;

• стиль, заданный в теге <style>, будет отменен, если в параметре style тега указан другой стиль;

• стиль, заданный в отдельном файле, будет отменен, если в теге <style> указано другое определение стиля.

Именно из-за такой структуры приоритетов таблицы стилей называют каскадными. То есть наименьший приоритет имеет стиль, описанный в отдельном файле, а самый высокий – стиль, указанный последним.

Ниже представлен пример, в котором четырьмя разными способами задан цвет текста абзаца.

Содержимое файла style.css:

```
p { color: red }
```

Содержимое файла test.html:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Приоритет применения стилей</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<link rel="stylesheet" type="text/css" href="style.css">
<style type="text/css">
p { color: blue }
</style>
</head>
<body>
<p style="color: green"><font color="yellow">Текст1</font></p><br>
```

* Текст2 *
</body>
</html>

Учитывая приоритет применения стилей, в этом примере к абзацу со словом "Текст1" будет применено форматирование, указанное параметром color тега <font>, то есть абзац будет желтого цвета. А абзац со словом "Текст2" будет иметь цвет, указанный в параметре style, то есть зеленый.

Кроме того, следует отметить, что стиль, заданный через идентификатор, будет иметь более высокий приоритет, чем стиль, заданный через класс. Например:

<style type="text/css">
#id1 { color: red }
.cls1 { color: blue }
</style> ...
* Текст1 *

В этом примере текст абзаца будет красного цвета, а не синего.

С помощью свойства !important можно изменить приоритет. Например, если изменить содержимое файла style.css на: *p { color: red !important }* можно переопределить все стили и абзац со словом "Текст2" будет иметь красный цвет, а не зеленый. Однако абзац со словом "Текст1" так и останется желтого цвета, так как цвет указан в параметре color тега <font>, а не задан через стиль.

5.4. Единицы измерения в CSS

Размеры в CSS можно задавать в абсолютных или относительных единицах.

Абсолютные единицы:

- пиксел (px);
- миллиметр (mm);
- сантиметр (cm);
- дюйм (in) – 1 in = 2.54 cm;
- пункт (pt) – 1 pt = 1/72 in;

• пика (pc) – 1 pc = 12 pt.

Относительные единицы:

• процент (%);

• высота текущего шрифта (em);

• высота буквы "x" текущего шрифта (ex).

Для значения 0 можно не указывать единицы измерения.

Цвет можно задать одним из следующих способов:

• именем цвета – blue, green и т.д. Например: *p { color: red }*

• значением вида #[R][G][B], где R – насыщенность красного, G – насыщенность зеленого и B – насыщенность синего в выбранном цвете. Значения задаются одинарными шестнадцатеричными числами от 0 до F. Например: *p { color: #F00 }*

• значением вида #[RR][GG][BB], где RR – насыщенность красного, GG – насыщенность зеленого и BB – насыщенность синего в выбранном цвете. В таком формате значения задаются двузначными шестнадцатеричными числами от 00 до FF. Например: *p { color: #FF0000 }*

• значением вида rgb([R], [G], [B]), где R, G и B – насыщенность красного, зеленого и синего цветов соответственно, которая задается десятичными числами от 0 до 255. Например: *p { color: rgb(255, 0, 0) }*

• значением вида rgb([R%], [G%], [B%]), где R%, G% и B% – насыщенность красного, зеленого и синего цветов, которая задается в процентах. Например: *p { color: rgb(100%, 0%, 0%) }*

Все представленные выше примеры задают красный цвет.

5.5. Форматирование шрифта

Каскадные таблицы стилей позволяют задать название, цвет и размер шрифта, а также его стиль и начертание. Кроме того, можно указать несколько названий шрифтов и одно из названий альтернативных семейств. Так как на компьютере пользователя может не оказаться нужного шрифта.

Атрибут font-family позволяет задать название шрифта. Например:
p { font-family: "Arial" }

Лучше указать названия нескольких альтернативных шрифтов через

запятую. Например: *p { font-family: "Verdana", "Tahoma" }*

Можно также указать одно из пяти типовых семейств шрифтов – serif, sans-serif, cursive, fantasy или monospace. Например:

p { font-family: "Verdana", "Tahoma", sans-serif }

Атрибут font-style позволяет задать стиль шрифта. Он может принимать следующие значения:

- normal – обычный шрифт. Например:

p { font-family: "Arial"; font-style: normal }

- italic – курсивный шрифт. Например:

p { font-family: "Arial"; font-style: italic }

- oblique – наклонный шрифт. Например:

p { font-family: "Arial"; font-style: oblique }

Атрибут font-size позволяет задать размер шрифта. Например:

.text1 { font-size: 12pt; font-family: "Arial" }

Можно указать абсолютную величину или одну из типовых констант – xx-small, x-small, small, medium, large, x-large или xx-large. Например:

.text1 { font-size: large; font-family: "Arial" }

Кроме того, можно указать относительную величину (например, значение в процентах) или одну из двух констант – larger или smaller. Например:

.text1 { font-size: 150%; font-family: "Arial" }

.text2 { font-size: smaller; font-family: "Arial" }

Атрибут color позволяет задать цвет шрифта. Например:

.text1 { font-size: 12pt; font-family: "Arial"; color: red }

.text2 { font-size: 12pt; font-family: "Arial"; color: #00FF00 }

.text3 { font-size: 12pt; font-family: "Arial"; color: rgb(255, 0, 0) }

Атрибут font-weight позволяет управлять "жирностью" шрифта и может принимать следующие значения:

- 100, 200, 300, 400, 500, 600, 700, 800, 900 – значение 100 соответствует самому бледному шрифту, а 900 – самому жирному. Например:

p { font-family: "Arial"; font-style: italic; font-weight: 700 }

- normal – обычный шрифт соответствует значению 400. Например:

p { font-family: "Arial"; font-style: italic; font-weight: normal }

• bold – полужирный шрифт соответствует значению 700. Например:

p { font-family: "Arial"; font-style: italic; font-weight: bold }

5.6. Форматирование текста

Для текстовых фрагментов можно также задать некоторые дополнительные параметры.

Атрибут letter-spacing задает расстояние между символами текста. Он может задаваться следующим образом:

• по умолчанию, т.е. принимать значение normal. Например:

p { letter-spacing: normal; font-style: italic; font-weight: normal }

• абсолютной величиной в поддерживаемых CSS единицах. Например:

p { font-size: 12pt; color: red; letter-spacing: 5mm }

Атрибут word-spacing задает расстояние между словами. Он может задаваться следующим образом:

• по умолчанию, т.е. принимать значение normal. Например:

p { word-spacing: normal; font-style: italic; font-weight: normal }

• абсолютной величиной в поддерживаемых CSS единицах. Например:

p { font-size: 12pt; color: red; word-spacing: 5mm }

Атрибут text-indent задает отступ "красной строки". Может задаваться абсолютная или относительная величина отступа. Например:

p { text-indent: 10mm; font-style: italic; font-weight: normal }

Атрибут line-height задает вертикальное расстояние между базовыми линиями двух строк. Он может задаваться следующим образом:

• по умолчанию, т.е. принимать значение normal. Например:

p { line-height: normal; font-style: italic; font-weight: normal }

• абсолютной или относительной величиной в поддерживаемых CSS единицах:

p { font-size: 12pt; color: red; font-family: "Arial"; line-height: 5mm }

Атрибут text-align задает горизонтальное выравнивание текста. Этот атрибут может принимать следующие значения:

• center – выравнивание по центру. Например:

* Абзац с выравниванием по центру *

• left – выравнивание по левому краю. Например:

* Абзац с выравниванием по левому краю *

• right – выравнивание по правому краю. Например:

* Абзац с выравниванием по правому краю *

• justify – выравнивание по ширине. Например:

* Абзац с выравниванием по ширине *

Атрибут vertical-align задает вертикальное выравнивание текста относительно элемента-родителя, например, ячейки таблицы. Он может принимать следующие значения:

• baseline – по базовой линии. Например:

td { font-size: 12pt; color: red; vertical-align: baseline }

• middle – по центру. Например:

td { font-size: 12pt; color: red; vertical-align: middle }

• top – по верхнему краю. Например:

td { font-size: 12pt; color: red; vertical-align: top }

• bottom – по нижнему краю. Например:

td { font-size: 12pt; color: red; vertical-align: bottom }

Атрибут text-decoration позволяет подчеркнуть, надчеркнуть или зачеркнуть текст. Он может принимать следующие значения:

• none – обычный текст (по умолчанию). Например:

* Текст *

• underline – подчеркнутый текст. Например:

* Подчеркнутый текст *

• overline – надчеркнутый текст. Например:

* Надчеркнутый текст *

• line-through – зачеркнутый текст. Например:

* Зачеркнутый текст *

• blink – мигающий текст. Например:

* Мигающий текст *

Атрибут text-transform позволяет изменить регистр символов. Он может принимать следующие значения:

• capitalize – делает первую букву каждого слова прописной;

- uppercase – преобразует все буквы в прописные;
- lowercase – преобразует все буквы в строчные;
- none – не делает никаких преобразований.

Пример:

```
<h1 style="text-transform: capitalize">заголовок из нескольких слов</h1>
<h1 style="text-transform: uppercase">заголовок2</h1>
<h1 style="text-transform: lowercase">ЗАГОЛОВОК3</h1>
```

Результат:

Заголовок Из Нескольких Слов

ЗАГОЛОВОК2

заголовок3

Атрибут white-space позволяет установить тип обработки пробелов. По умолчанию несколько пробелов подряд выводятся в окне веб-браузера как один пробел. Атрибут может принимать следующие значения:

- normal – текст выводится стандартным образом. Например:

```
<p style="white-space: normal">
Строка 1
Строка 2
</p>
```

Результат в окне веб-браузера:

Строка 1 Строка 2

- pre – сохраняются все пробелы и переносы строк. Например:

```
<p style="white-space: pre">
Строка 1
Строка 2
</p>
```

Результат в окне веб-браузера:

Строка 1

Строка 2

- nowrap – переносы строк в HTML-коде игнорируются. Если внутри строки содержится тег
, то он вставляет перенос строки. Например:

```
<p style="white-space: nowrap">
```

Строка 1
*Строка 2
*
Строка 3
</p>

Результат в окне веб-браузера:

Строка 1 Строка 2
Строка 3

5.7. Поля и отступы

Любой элемент веб-страницы занимает в окне веб-браузера некоторую прямоугольную область. Причем эта область имеет как внутренние, так и внешние отступы. Внутренний отступ – это расстояние между элементом страницы и реальной или воображаемой границей области. Внешний отступ – это расстояние между реальной или воображаемой границей и другим элементом веб-страницы, т.е. между границей и крайней точкой внешнего отступа другого элемента веб-страницы.

С помощью атрибутов margin-left, margin-right, margin-top и margin-bottom можно задать поля одного элемента веб-страницы от другого. Может быть задано абсолютное или относительное значение. Более того, атрибуты могут иметь отрицательные значения. Эти атрибуты означают:

- margin-left – поле слева. Например: *body { margin-left: 0 }*
- margin-right – поле справа. Например: *body { margin-right: 5% }*
- margin-top – поле сверху. Например: *body { margin-top: 15mm }*
- margin-bottom – поле снизу. Например: *body { margin-bottom: 20px }*

Убрать все внешние поля можно несколькими способами. Например:
<body style="margin-left: 0; margin-right: 0; margin-top: 0; margin-bottom: 0">
или body { margin-left: 0; margin-right: 0; margin-top: 0; margin-bottom: 0 }

С помощью атрибута margin можно задать все внешние поля за один раз: margin: <top> <right> <bottom> <left>. Например:
body { margin: 15mm 5% 20px 0 }

Для совпадающих значений допускается и более короткая запись. Например: *body { margin: 0 }*

С помощью атрибутов padding-left, padding-right, padding-top и padding-bottom можно задать отступы от элемента веб-страницы до его рамки (если она есть). Например, ими задается расстояние между текстом и рамкой ячейки таблицы. Может быть задано абсолютное или относительное значение:

- padding-left – отступ слева. Например: *td { padding-left: 0 }*
- padding-right – отступ справа. Например: *td { padding-right: 50px}*
- padding-top – отступ сверху. Например: *td { padding-top: 15mm}*
- padding-bottom – отступ снизу. Например: *td { padding-bottom: 20px}*

С помощью атрибута padding можно задать все внутренние отступы за один раз: padding: <top> <right> <bottom> <left>. Например:
td { padding: 15mm 50px 20px 0 }

Совпадающие отступы можно задать и короче: *td { padding: 5px }*

5.8. Рамки (границы)

Содержимое области, которую занимает любой элемент веб-страницы в окне веб-браузера, может быть окружено рамками.

С помощью атрибутов border-left-style (левая линия), border-right-style (правая линия), border-top-style (верхняя линия) и border-bottom-style (нижняя линия) можно задать тип линий рамки. Эти атрибуты могут принимать следующие значения:

- none – линия не отображается;
- solid – сплошная линия;
- dotted – пунктирная линия;
- dashed – штриховая линия;
- double – двойная линия;
- groove – вдавленная рельефная линия;
- ridge – выпуклая рельефная линия;
- inset – весь блок элемента отображается вдавленным;
- outset – весь блок элемента отображается выпуклым.

Ниже представлен пример, в котором указан стиль линии рамки для разных элементов веб-страницы.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Тип рамки</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<style type="text/css">
table { border-left-style: dashed; border-right-style: dotted;
border-top-style: solid; border-bottom-style: groove }
td { border-left-style: none; border-right-style: none;
border-top-style: none; border-bottom-style: none; text-align: center }
caption { border-top-style: solid }
p { border-left-style: dotted; border-right-style: dotted;
border-top-style: dotted; border-bottom-style: dotted }
</style>
</head>
<body>
<table width="200">
<caption>Заголовок таблицы</caption>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</table>
<p>Текст в пунктирной рамке</p>
</body>
</html>
```

Атрибуты, определяющие стили линий рамки, могут быть объединены

в одном атрибуте border-style: border-style: <top> <right> <bottom> <left>. Если все значения совпадают, можно указать это значение один раз.

С помощью атрибутов border-left-width (левая линия), border-right-width (правая линия), border-top-width (верхняя линия) и border-bottom-width (нижняя линия) можно задать толщину линии рамки. Может быть задано абсолютное значение. Например:

table { border-left-width: 5px; border-right-width: 5px;
border-top-width: 0; border-bottom-width: 10px }

Также можно указать одно из предопределенных значений:

- thin – тонкая линия;
- medium – средняя толщина линии;
- thick – толстая линия.

Например:

table { border-left-width: medium; border-right-width: medium;
border-top-width: thin; border-bottom-width: thick }

Эти атрибуты могут быть объединены в одном атрибуте border-width: border-width: <top> <right> <bottom> <left>. Если значения совпадают, можно указать их один раз.

С помощью атрибутов border-left-color (левая линия), border-right-color (правая линия), border-top-color (верхняя линия) и border-bottom-color (нижняя линия) можно задать цвет линий рамки. Например:

table { border-left-color: red; border-right-color: #000080;
border-top-color: green; border-bottom-color: black }

Как и в случае border-style и border-width, эти атрибуты можно объединить в один атрибут border-color.

Если атрибуты рамки одинаковы для всех ее сторон, можно задавать их в одном атрибуте border: border: <стиль> <толщина> <цвет>

Поскольку значение атрибута однозначно определяет к какому именно компоненту он относится, то их можно указывать в произвольном порядке. Например: *td { border: red thin solid }*

5.9. Фон элемента

Каскадные таблицы стилей позволяют задать цвет фона или использовать в качестве фона изображение. Для фонового рисунка можно задать начальное положение и указать будет ли рисунок прокручиваться вместе с содержимым веб-страницы. Кроме того, можно установить как фоновый рисунок будет повторяться (по вертикали и/или по горизонтали).

С помощью атрибута background-color можно задать цвет фона. Например:

body { background-color: green }

td { background-color: silver }

В качестве значения атрибута можно использовать слово transparent, в этом случае фон будет прозрачным. Например:

td { background-color: transparent }

С помощью атрибута background-image можно задать URL-адрес изображения, которое будет использовано в качестве фонового рисунка. Может быть указан абсолютный или относительный URL-адрес (относительно местоположения таблицы стилей, а не документа). Например:

body { background-image: url(foto1.gif) }

В качестве значения атрибута можно использовать слово none, в этом случае фоновая заливка будет отключена. Например:

body { background-image: none }

Атрибут background-repeat задает режим повтора фонового рисунка. Он может принимать следующие значения:

- repeat – рисунок повторяется и по вертикали, и по горизонтали (по умолчанию). Например:

body { background-image: url(foto1.gif); background-repeat: repeat }

- repeat-x – рисунок повторяется по горизонтали. Например:

body { background-image: url(foto1.gif); background-repeat: repeat-x }

- repeat-y – рисунок повторяется по вертикали. Например:

body { background-image: url(foto1.gif); background-repeat: repeat-y }

- no-repeat – рисунок не повторяется. Например:

body { background-image: url(foto1.gif); background-repeat: no-repeat }

Атрибут background-attachment определяет будет ли фоновый рисунок прокручиваться вместе с содержимым веб-страницы. Он может принимать следующие значения:

• scroll – фоновый рисунок прокручивается вместе с содержимым страницы (по умолчанию). Например:

body { background-image: url(foto1.gif);
background-repeat: no-repeat; background-attachment: scroll }

• fixed – фоновый рисунок не прокручивается. Например:

body { background-image: url(foto1.gif);
background-repeat: no-repeat; background-attachment: fixed }

Атрибут background-position задает начальное положение фонового рисунка. В качестве значений атрибута через пробел указываются координаты в абсолютных единицах или процентах. Например:

body { background-image: url(foto1.gif); background-repeat: no-repeat;
background-attachment: fixed; background-position: 50% 50% }

Кроме того, могут быть указаны следующие значения:

• left – выравнивание по левому краю;
• right – выравнивание по правому краю;
• center – выравнивание по центру;
• top – выравнивание по верхнему краю;
• bottom – выравнивание по нижнему краю.

Например:

body { background-image: url(foto1.gif); background-repeat: no-repeat;
background-attachment: fixed; background-position: left center }

Атрибут background используется для одновременного указания значений атрибутов background-color, background-image, background-position, background-repeat и background-attachment. Например:

body { background: green url(foto1.gif) no-repeat fixed left center }
body { background: green }

Во втором примере указан только цвет фона, а остальные атрибуты не указаны. Эти атрибуты получат значения по умолчанию.

Кроме того, поскольку значение атрибута однозначно определяет к

какому именно компоненту он относится, то их можно указывать в произвольном порядке.

5.10. Списки

Задать параметры списка можно не только с помощью параметров тегов, но и с помощью атрибутов стиля. Более того, каскадные таблицы стилей позволяют использовать в качестве маркера списка любое изображение.

Атрибут list-style-type задает вид маркера списка. Он может принимать следующие значения:

• disc – выводит маркер в виде круга с заливкой. Например: *ul { list-style-type: disc }*

• circle – выводит маркер в виде круга без заливки. Например: *ul { list-style-type: circle }*

• square – выводит маркер в виде квадрата с заливкой. Например: *ul { list-style-type: square }*

• decimal – строки нумеруются арабскими цифрами. Например: *ol { list-style-type: decimal }*

• lower-roman – строки нумеруются строчными римскими цифрами. Например: *ol { list-style-type: lower-roman }*

• upper-roman – строки нумеруются прописными римскими цифрами. Например: *ol { list-style-type: upper-roman }*

• lower-alpha – строки нумеруются строчными латинскими буквами. Например: *ol { list-style-type: lower-alpha }*

• upper-alpha – строки нумеруются прописными латинскими буквами. Например: *ol { list-style-type: upper-alpha }*

• none – не помечает пункты списка. Например: *ol { list-style-type: none }*

Атрибут list-style-image задает URL-адрес изображения, которое будет использовано в качестве маркера списка. Относительные адреса указываются относительно расположения таблицы стилей, а не HTML-документа. Например: *ol { list-style-image: url(foto1.gif) }*

Атрибут list-style-position позволяет задать более компактное отображение списка. Может принимать следующие значения:

• outside – по умолчанию. Маркер отображается отдельно от текста.

Например: *ol { list-style-position: outside }*

• inside – более компактное отображение списка. Маркер входит в состав текста. Например: *ol { list-style-position: inside }*

5.11. Вид курсора

Атрибут cursor задает форму курсора мыши при наведении на элемент страницы. Может принимать следующие значения:

• auto – веб-браузер сам определяет форму курсора мыши. Например: *p { cursor: auto }*

• crosshair – курсор в виде креста. Например: *p { cursor: crosshair }*

• default – курсор в виде стрелки (курсор по умолчанию). Например: *p { cursor: default }*

• pointer – курсор в виде руки. Например: *p { cursor: pointer }*

• move – курсор в виде стрелки, указывающей во все направления. Например: *p { cursor: move }*

• n-resize, ne-resize, nw-resize, s-resize, se-resize, sw-resize, eresize, w-resize – курсор в виде стрелок, показывающих направление. Например: *p { cursor: n-resize }*

• text – курсор в виде буквы Т. Например: *p { cursor: text }*

• wait – курсор в виде песочных часов. Например: *p { cursor: wait }*

• progress – курсор в виде стрелки с песочными часами. Например: *p { cursor: progress }*

• help – курсор в виде стрелки с вопросительным знаком. Например: *p { cursor: help }*

5.12. Псевдостили гиперссылок

Большинство веб-браузеров позволяют отобразить посещенные и непосещенные ссылки разными цветами. Достигается это при помощи следующих предопределенных стилей или псевдостилей:

• a:link – определяет вид непосещенной ссылки;

• a:visited – определяет вид посещенной ссылки;

• a:active – определяет вид активной ссылки;

• a:hover – определяет вид ссылки, на которую указывает курсор мыши.

При использовании этих псевдостилей в коде до и после двоеточия не должно быть пробелов.

Псевдостили аналогичны параметрам link, vlink и alink тега <body>. Например, строка *<body link="#000000" vlink="#000080" alink="#FF0000">* эквивалентна такому заданию стиля:

```
a:link { color: #000000 }
a:visited { color: #000080 }
a:active { color: #FF0000 }
```

С помощью псевдостилей можно изменять не только цвет ссылки, но и задавать, будет ли ссылка подчеркнута. Например:

```
a:link { color: red; text-decoration: underline }
a:visited { color: blue; text-decoration: underline }
a:active { color: green; text-decoration: none }
a:hover { color: lime; text-decoration: none }
```

Кроме того, можно применить стиль гиперссылок не только ко всему документу, но и к определенному классу. Например:

```
a.link1:link { color: black; text-decoration: none }
a.link1:visited { color: blue; text-decoration: none }
a.link1:active { color: red; text-decoration: underline }
a.link1:hover { color: red; text-decoration: underline }
```

Ниже представлен пример использования псевдостилей гиперссылок в html-документе.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Псевдостили гиперссылок</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<style type="text/css">
a:link { color: red; text-decoration: underline }
a:visited { color: blue; text-decoration: underline }
```

```
a:active { color: green; text-decoration: none }
a:hover { color: lime; text-decoration: none }
a.link1:link { color: black; text-decoration: none }
a.link1:visited { color: blue; text-decoration: none }
a.link1:active { color: red; text-decoration: underline }
a.link1:hover { color: red; text-decoration: underline }
</style>
</head>
<body>
<p>
<a href="doc1.html">Ссылка1</a><br>
<a href="doc2.html" class="link1">Ссылка2</a>
</p>
</body>
</html>
```

Практические задания

1. Применить свойства CSS для форматирования произвольного фрагмента текста.

Контрольные вопросы

1. Что такое стиль?
2. Каково функциональное назначение каскадирования?
3. Какими способами можно подключать таблицы стилей?
4. Какие типы селекторов можно использовать при форматировании веб-страницы?
5. Какие свойства CSS имеют аналоги в HTML, а какие являются уникальными?
6. Опишите виды списков в CSS.

6. РАЗРАБОТКА WEB-САЙТОВ С ИСПОЛЬЗОВАНИЕМ БЛОЧНОЙ ВЕРСТКИ

6.1. Основные сведения

Первоначально слои ввела компания Netscape, включив в свой браузер поддержку тега <LAYER>. Этот тег позволял прятать/показывать текущее содержимое, устанавливать положение относительно окна браузера, накладывать один слой поверх других и загружать данные из файла в содержимое слоя. Несмотря на столь впечатляющий набор возможностей, тег <LAYER> не был включен в спецификацию HTML, и так и остался лишь расширением браузера Netscape.

Однако необходимость в указанных возможностях, уже назрела, и в конце 1996 года синтаксис для работы со слоями был разработан и одобрен в рабочем проекте консорциума "CSS Positioning (CSS-P)". Основная нагрузка ложилась на стили, с их помощью можно было управлять видом любого элемента. К сожалению, объектные модели браузеров для доступа к элементам существенно отличались, поэтому приходилось писать достаточно сложный код, который бы учитывал эти различия.

В настоящее время разработчики популярных браузеров стали придерживаться спецификаций HTML и CSS, что позволило сократить затраты времени на отладку сайта в разных браузерах. Тем не менее, различия в подходах у браузеров существуют и при их возникновении разработчики придерживаются следующих форм работы:

1. Если наблюдаются небольшие различия в отображении сайта в разных браузерах, то эти отличия игнорируются. Но сайт в любом случае должен отображаться корректно и без ошибок.

2. Если наблюдаются существенные различия в отображении сайта в разных браузерах, то для их устранения можно применять хаки. Хак – это набор приемов, когда определенному браузеру "подсовывают" код, который понимается только этим браузером, а остальными игнорируется. Несмотря на то, что хаки работают, использовать их следует ограниченно или вообще обходиться без них, поскольку хаки снижают универсаль-

ность кода и для модификации параметров одного элемента приходится вносить изменения одновременно в разных местах.

Но все-таки для устранения различий в отображении сайта лучше изначально при создании сайта придерживаться спецификации CSS. Несмотря на то, что пока браузеры не в полной мере сами ее поддерживают, они прогрессируют в направлении полной поддержки различных спецификаций (HTML, CSS, DOM), т.е. будущие версии браузеров будут унифицированы и один и тот же сайт будет отображаться корректно.

Блочная верстка сайта заключается в конструктивном использовании тегов <DIV>, <SPAN> и др., а также стилей. При этом следует придерживаться следующих принципов:

1. Разделение содержимого и оформления. Код HTML должен содержать только теги разметки и теги логического форматирования, а любое оформление выносится за пределы кода в стили. Такой подход позволяет независимо управлять видом элементов страницы и ее содержимым. Благодаря этому над сайтом может работать одновременно и автономно несколько человек, затрачивая при этом меньше времени на разработку сайта.

2. Активное применение тега DIV. При блочной верстке большое значение имеет универсальный тег <DIV>, который выполняет множество функций. Благодаря этому тегу HTML-код делится на ряд четких наглядных блоков, поэтому такая верстка называется блочной. Код при этом получается более компактным, чем при табличной верстке, к тому же поисковые системы его лучше индексируют.

3. Использование таблиц только для представления табличных данных, например, для наглядного отображения числовых данных и др.

4. Ограничение высоты блоков высотой контента. В таблице соседние ячейки взаимосвязаны, поэтому высота у них одна, независимо от объема информации. Это хорошо видно, если залить фон ячеек разным цветом. Блоки же в каком-то смысле являются независимыми друг от друга, поэтому высота блока определяется его содержимым. Вид документа при этом будет отличаться от его табличного аналога (рис. 6.1).

Рис. 6.1. Страница, созданная с помощью блоков

В случае необходимости высоту блока можно установить через стили, используя для этого разные единицы (проценты, пикселы, дюймы и т.д.). Но если содержимое блока превышает заданную высоту, то браузеры по-разному его интерпретируют: одни расширяют высоту блока под новый контент, а другие – оставляют высоту исходного блока такой же и накладывают контент поверх блока.

5. Блочная верстка. Поскольку блоки в большинстве случаев являются независимыми друг от друга, они могут добавляться или удаляться в макете веб-страницы как отдельные блоки. Блоки допустимо вкладывать один в другой для формирования сложной конструкции.

6. Расположение колонок. По умолчанию содержимое контейнеров <DIV> на веб-странице располагается по вертикали: вначале идет один блок, ниже располагается следующий и т.д. При создании колонок необходимо располагать блоки рядом по горизонтали, для чего применяется несколько методов. Одним из наиболее распространенных является использование стилевого параметра float. Хотя его основная функция состоит в создании обтекания вокруг элемента, он может быть использован и для создания колонок. Но здесь следует учесть одну особенность. При уменьшении окна браузера до некоторой критической ширины колонки перестают располагаться горизонтально и располагаются друг под другом по вертикали. В таком случае можно либо оставить все как есть, либо использовать другие методы формирования колонок через блоки.

7. Блочные элементы. Блочным называется элемент, который отображается на веб-странице в виде прямоугольника. Такой элемент занимает всю доступную ширину, высота элемента определяется его содержимым, и он всегда начинается с новой строки. К блочным элементам относятся контейнеры <DIV>, <H1>, <P> и др.

Допускается вкладывать один блочный элемент в другой, а также размещать внутри них встроенные элементы (например, <SPAN>). Не рекомендуется добавлять внутрь встроенных элементов блочные. Например:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Блочные элементы</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<div><p>Lorem ipsum dolor sit amet...</p></div>
<h2><a href="linkl.html">Ut wisi enim ad minim veniam</a></h2>
</body>
</html>
```

В примере, приведенном выше, параграф (тег <P>) вставляется внутрь контейнера <DIV>, а ссылка (тег <A>) – в заголовок <H2>. Однако неприемлемым является вставка элемента <H2> в контейнер <A>, поскольку тег <A> не относится к блочным элементам.

6.2. Ширина блочных элементов

По умолчанию ширина блока определяется автоматически и занимает все доступное пространство. Под этим подразумевается следующее. Если, например, в коде документа присутствует один тег <DIV>, он занимает всю свободную ширину окна браузера и ширина блока будет равна 100 %. Но если поместить один тег <DIV> внутрь другого, то ширина внутреннего тега определяется относительно его родительского элемента, т.е. внеш-

него контейнера.

Некоторые браузеры достаточно свободно трактуют понятие ширины, хотя в спецификации CSS четко указано, что ширина складывается из суммы следующих параметров: ширины самого блока (width), отступов (margin), полей (padding) и границ (border) (рис. 6.2).

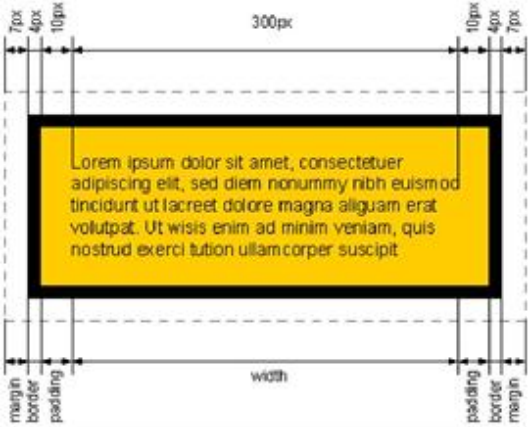


Рис. 6.2. Ширина блока

Ниже приведен пример кода, создающего блок шириной 342 px, в котором присутствуют все компоненты, описанные выше:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Ширина</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<style type="text/css">
DIV { width: 300px; /* Ширина слоя */
margin: 7px; /* Значение отступов */
padding: 10px; /* Поля вокруг текста */
border: 4px solid black; /* Параметры границы */
background: #fcO; /* Цвет фона */ }
</style>
```

```
</head>
<body>
<div>Lorem ipsum dolor sit amet...</div>
</body>
</html>
```

В том случае когда !DOCTYPE в коде не указан, браузер Internet Explorer за ширину всего блока принимает значение параметра width. По умолчанию ширина блока задается как auto, это позволяет вписывать блок в окно браузера, не принимая в расчет значения установленных полей. Если изменить ширину на 100 %, то при добавлении значения отступов, полей или границ обязательно появится горизонтальная полоса прокрутки.

Ниже приведен пример кода, создающего три блока, ширина которых определяется в процентах.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Ширина</title>
<meta http-equiv="Content-Type" content="text/html; charset-windows-1251">
<style type="text/css">
#layer1 { width: 100%; /* Ширина первого слоя */
padding: 10px; /* Поля вокруг текста */
background: #fc0; /* Цвет фона */ }
#1ayer2 { width: 100%; /* Ширина второго слоя */
background: #cc0; /* Цвет фона */ }
#1ayer2 P { padding: 10px; /* Поля вокруг параграфа */ }
#1ayer3 { background: #3ca; /* Цвет фона третьего слоя */
padding: 10px; /* Поля вокруг текста */ }
</style>
</head>
<body>
<div id="layer1">Lorem ipsum dolor sit amet...</div>
```

* Lorem ipsum dolor sit amet... *

* Lorem ipsum dolor sit amet... *

</body>

</html>

Созданные блоки представлены на рис. 6.3.

Рис. 6.3. Варианты установки ширины блока в процентах

Ширина первого блока (layer1) в данном примере установлена как 100 %, что приводит к отображению горизонтальной полосы прокрутки. Для второго блока (layer2) ширина также задана 100 %, но поля определяются для внутреннего параграфа (тег <P>). За счет этого ширина блока во всех браузерах будет одинаковой. К третьему блоку (layer3) вообще не применяется параметр width, поэтому он определяется по умолчанию как auto. В этом случае блок будет занимать всю ширину окна браузера без горизонтальных полос прокрутки.

Способ установки ширины зависит от применяемого макета и выбора разработчика, но в любом случае нужно учитывать особенности блочных элементов и создавать универсальный код.

6.3. Высота блочных элементов

С высотой блочных элементов дело обстоит аналогично ширине, т.е. браузер Internet Explorer (а также Opera) за высоту блока принимает значение параметра height, а Firefox добавляет к нему еще значение параметров margin, padding и border. Если высота блока не установлена явно, то она вычисляется автоматически, исходя из объема содержимого.

6.4. Цвет фона

Цвет фона элемента проще всего устанавливать через универсальный параметр background. Фоновым цветом при этом заливается область, которая определяется значениями атрибутов width, height и padding (рис. 6.4).

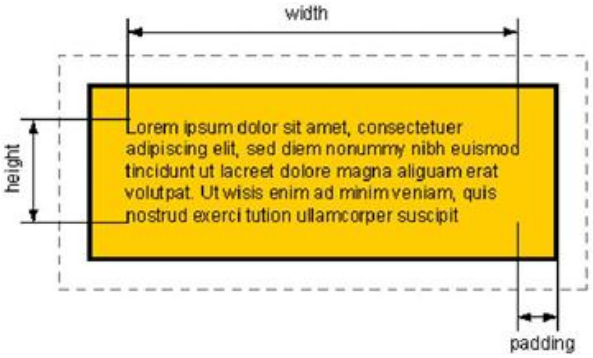


Рис. 6.4. Область блока, которая заполняется фоновым цветом

6.5. Границы

Из-за различий в подходах браузеров к формированию блочных элементов наблюдаются и различия при отображении границ. Например, браузер Internet Explorer проводит рамку внутри блока, а Firefox – снаружи. Но если использовать фоновую заливку, то результат будет противоположным (рис. 6.5), потому что Firefox (Opera) цвет фона устанавливает по внешнему краю границы, а Internet Explorer – по внутреннему.

а) Internet Explorer

б) Firefox

в) Opera

Рис. 6.5. Отображение рамки элемента в разных браузерах

Различия в подходах браузеров к отображению границ заметны только на цветном фоне и пунктирных линиях. Для сплошной рамки вид блока в браузерах будет практически одинаковым.

6.6. Встроенные элементы

Встроенными называются элементы веб-страницы, являющиеся непосредственной частью другого элемента. К встроенным элементам относятся теги <SPAN>, <A>, <Q>, <CODE> и др. В основном они используются для изменения вида текста или его логического выделения. Отличия между блочными и встроенными элементами состоят в следующем:

1. Встроенные элементы могут содержать только данные или другие встроенные элементы, а в блочные элементы допустимо вкладывать другие блочные элементы, встроенные элементы, а также данные, т.е. встроенные элементы не могут содержать блочные элементы.

2. Блочные элементы всегда начинаются с новой строки, а для встроенных элементов это правило не является обязательным.

3. Блочные элементы занимают всю доступную ширину (например, окна браузера), а ширина встроенных элементов равна их содержимому плюс значения отступов, полей и границ.

Встроенные элементы можно превращать в блочные с помощью свойства display и его значения block. Также возможно и обратное действие через аргумент inline.

Основные значения свойства display приведены в табл. 6.1 (полный список можно найти по адресу http://htmlbook.ru/css/display.html).

Таблица 6.1 – Основные значения свойства display

Аргумент	**Описание**
1	2
block	Элемент показывается как блочный. Применение этого значения для встроенных элементов, например, для тега <SPAN>, присваивает ему некоторые свойства блоков – происходит

	перенос строк в начале и в конце содержимого

Продолжение таблицы 6.1

1	2
Inline	Элемент отображается как встроенный. Например, использование блочных тегов, таких как <DIV> и <P>, автоматически создает перенос и показывает содержимое этих тегов с новой строки. Аргумент inline отменяет эту особенность, поэтому содержимое блочных элементов начинается с того места, где окончился предыдущий элемент
inline- block	Это значение генерирует блочный элемент, который обтекается другими элементами веб-страницы подобно встроенному элементу. Фактически такой элемент по своему действию похож на встраиваемые элементы (например, тег <IMG>). При этом его внутренняя часть форматируется как блочный элемент, а сам элемент – как встроенный
list-item	Элемент выводится как блочный и добавляется маркер списка
None	Временно удаляет элемент из документа. Занимаемое им место не резервируется и веб-страница формируется так, как будто элемента и не было. Изменить значение параметра и сделать его вновь видимым можно с помощью скриптов, обращаясь к свойствам через объектную модель. В этом случае происхоит переформатирование данных на странице с учетом вновь добавленного элемента

Встроенные элементы применяются не только для управления видом текста, но также и при верстке веб-страниц, например, для изменения положения блоков. Ниже представлен пример кода, позволяющего наложить рисунок поверх блока с текстом.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset-windows-1251">
<title>Встроенные элементы</title>
```

```
<style type="text/css">
.send { background: #d6d3a2; /* Цвет фона */
padding: 5px; /* Поля вокруг текста */
padding-left: 45px; /* Отступ слева */ }
.pic { position: relative; /* Относительное позиционирование */
top: –20px; /* Смещаем слой вверх */
left: 3px; /* Сдвигаем слой вправо */ }
</style>
</head>
<body>
<div class="send">Отправить ссылку на эту статью другу</div>
<div><span class="pic"><img src="mail.gif" width="45" height="38"
alt=""></span></div>
</body>
</html>
```

Результат представлен на рис. 6.6.

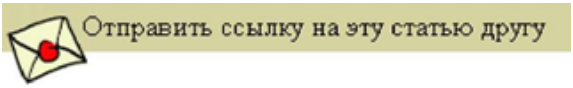


Рис. 6.6. Положение рисунка относительно текста

За счет того, что тег <SPAN> не начинается с новой строки, рисунок и текст в данном примере находятся на одной линии.

6.7. Плавающие элементы

Плавающими называются такие элементы, которые обтекаются по контуру другими объектами веб-страницы, например, текстом. Плавающие элементы достаточно активно применяются при верстке и в основном служат для: задания обтекания рисунков текстом; создания врезок; расположения блоков по горизонтали (добавления колонок). Все это выполняет один стилевой параметр float.

6.8. Обтекание рисунков текстом

Существуют разные способы объединения текста и рисунков. По-

умолчанию рисунок выравнивается по левому краю, а текст обтекает его по контуру.

6.9. Создание врезок

Врезкой называется блок с рисунками и текстом, который встраивается в основной текст. Врезка обычно располагается по левому или правому краю текстового блока, а основной текст обтекает ее с других сторон.

Чтобы врезка выделялась в тексте, для нее обычно задают фоновый цвет и добавляют рамку. При создании врезок следует обязательно указывать их ширину с помощью параметра width. Иначе ширина врезки может превысить ширину блока.

6.10. Расположение блоков по горизонтали

По умолчанию блоки выстраиваются по вертикали один под другим, но при помощи свойства float их можно расположить рядом по горизонтали. Для этого нужно установить ширину блоков и задать для них атрибут float.

Поскольку для второго блока также применяется обтекание, то добавленный ниже текст помещается справа от блока. Избежать этого можно, используя параметр clear, который отменяет действие свойства float.

Создание колонок при помощи float имеет ряд особенностей. Первая, как уже упоминалась, состоит в том, что после плавающих элементов следует добавлять элемент со свойством clear, который выключает обтекание. Это необходимо в том случае, если предполагается использовать нижележащие блоки. Вторая особенность связана с представлением плавающих блоков. Если окно браузера уменьшить до определенного предела, то блоки перемещаются по вертикали.

6.11. Выравнивание блока по центру

Веб-документ, отображаемый в окне браузера, может изменять свои размеры в зависимости от настроек операционной системы, типа монитора, установленного разрешения и т.д. Использование выравнивания позволяет проигнорировать указанную особенность и располагать элемент у края окна или по его центру. Выровнять блоки можно либо с помощью

отступов, либо с помощью позиционирования.

6.12. Использование отступов

Если добавить отступ к блоку слева с помощью параметра margin-left, то визуально блок сместится на указанное значение вправо. Зная ширину блока, его можно сместить так, чтобы блок располагался по центру веб-страницы. Для чего от 100 %, составляющих общую доступную ширину, надо отнять ширину блока в процентах и полученное значение разделить пополам. Результат и будет значением параметра margin-left. Пример кода, осуществляющего выравнивание блока по центру с помощью параметра margin-left представлен ниже.

```
<style type="text/css">
#centerLayer { margin-left: 30%; /* Отступ слева */
width: 40%; /* Ширина слоя */
background: #fc0; /* Цвет фона */
padding: 10px; /* Поля вокруг текста */ }
</style>
</head>
<body>
<div id="centerLayer"> Lorem ipsum dolor sit amet, consectetuer adipiscing
elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna
aliguam erat volutpat. </div>
```

Можно вообще не указывать ширину, а регулировать ее с помощью одинаковых значений отступов слева и справа (margin-left и margin- right).

Следующий способ более универсален и уже не зависит от того, какие единицы измерения используются для установки ширины. Для этого требуется задать отступ слева и справа для блока равным auto через стилевые атрибуты margin-left и margin-right или универсальное свойство margin (см. пример ниже).

```
<style type="text/css">
#centerLayer { width: 400px; /* Ширина слоя в пикселах */
margin: 0 auto; /* Отступ слева и справа */
background: #fc0; /* Цвет фона */
```

padding: 10px; / Поля вокруг текста */ }*
</style>
</head>
<body>
* Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat volutpat. *

В данном примере ширина блока устанавливается 400 пикселов и блок выравнивается по центру с помощью значения 0 auto параметра margin. Первый аргумент устанавливает нулевой отступ одновременно сверху и снизу от блока, а второй аргумент выравнивает блок по центру окна браузера.

6.13. Абсолютное позиционирование блока

При абсолютном позиционировании координаты блока вычисляются относительно левого верхнего угла окна родительского элемента или браузера, если родителя нет. Блок, заданный с помощью абсолютного позиционирования, может располагаться под основным текстом или поверх него. Положение определяется с помощью стилевого атрибута z-index и позволяет гибко управлять положением блока по условной z-оси. Таким способом удобно выводить на веб-странице различные подсказки, всплывающие окна, рекламу или плавающие меню.

Вначале следует указать ширину и высоту блока с помощью параметров width и height. Размеры можно задавать в пикселах, процентах или других единицах. Ширину, например, можно определить в процентах, а высоту в пикселах. Эта особенность делает предлагаемый метод размещения блока по центру наиболее универсальным.

На следующем этапе следует задать абсолютное позиционирование блока через аргумент position: absolute. Положение блока следует определить как 50 % по горизонтали и вертикали с помощью свойств left и top. Эти значения остаются неизменными независимо от используемых единиц измерения.

Так как координаты блока рассчитываются от его левого верхнего угла, для точного выравнивания следует добавить параметры margin-left и margin-top с отрицательными значениями. Их величина должна быть равна половине ширины блока (для margin-left) и половине высоты блока (для margin-top).

Чтобы высота блока не изменялась из-за его контента, следует включить параметр overflow: auto, который добавляет полосы прокрутки, если в них возникнет необходимость, высота блока при этом останется неизменной. Пример кода для выравнивания блока по центру с помощью абсолютного позиционирования представлен ниже.

```
<style type="text/css">
#centerLayer { position: absolute; /* Абсолютное позиционирование */
width: 400px; /* Ширина слоя в пикселах */
height: 300px; /* Высота слоя в пикселах */
left: 50%; /* Положение слоя от левого края */
top: 50%; /* Положение слоя от верхнего края */
margin-left: –200px; /* Отступ слева */
margin-top: –150px; /* Отступ сверху */
background: #fc0; /* Цвет фона */
border: solid 1px black; /* Параметры рамки вокруг */
padding: 10px; /* Поля вокруг текста */
overflow: auto; /* Добавление полосы прокрутки */ }
</style>
</head>
<body>
<div id="centerLayer">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy
nibh euismod tincidunt ut lacreet dolore magna aliguam erat volutpat.
</div>
```

Ширина и высота блока напрямую связаны с отступами слева и сверху, если необходимо установить значение одного из параметров в процентах, соответственно, изменится и запись другого параметра. Эта особен-

ность позволяет применять любые единицы измерения, а не ограничиваться только одной формой записи, что делает код подходящим практически для всех случаев.

В качестве справочной информации в табл. 6.2 приведено описание видов позиционирования блоков.

Таблица 6.2 – Виды позиционирования блоков

Аргумент	Описание
absolute	Указывает, что элемент абсолютно позиционирован. Положение элемента задается атрибутами left, top, right и bottom относительно края окна браузера
Fixed	По своим свойствам это значение аналогично аргументу absolute, но в отличие от него привязывается к указанной параметрами left, top, right и bottom точке на экране и не изменяет своего положения даже при пролистывании веб-страницы
Relative	Положение элемента устанавливается относительно его исходного местоположения. Добавление атрибутов left, top, right и bottom изменяет позицию элемента и сдвигает его в сторону относительно первоначального местоположения, в зависимости от используемого параметра
Static	Элементы отображаются как обычно. Использование параметров left, top, right и bottom не приводит к каким-либо результатам

6.14. Резиновый дизайн. Двухколонный макет

"Резиновым дизайном" называется структура веб-страницы, которая автоматически подстраивается под определенную ширину окна браузера пользователя. Основным преимуществом такого макета является то, что при грамотной верстке резиновый макет позволяет достичь большой универсальности в отображении независимо от ширины окна браузера и разрешения экрана пользователя. Среди недостатков резинового макета можно отметить различия в отображении сайтов в разных браузерах, в связи с чем повышается сложность верстки веб-страниц. Поэтому лучше использовать

гибрид фиксированного и резинового макета или гибридную верстку.

При верстке веб-страниц наиболее популярным является двухколонный макет, при этом одна колонка содержит набор ссылок для навигации по сайту, а вторая, более широкая – контент. Впрочем, хотя такая схема и является традиционной, это не означает, что ее следует придерживаться в любом случае. Использование всей ширины окна позволяет более эффективно задействовать площадь веб-страницы. Так что можно добавить еще несколько колонок. Это зависит исключительно от имеющегося объема информации. Однако в этом случае возможен и обратный эффект: чем больше материала будет на странице, тем больше будет рассеиваться внимание посетителя, и тем сложнее ему будет ориентироваться на сайте.

Создать двухколонный макет можно двумя способами. Первый подход использует параметр float, позволяющий состыковывать один блок с другим по горизонтали, а второй основан на задании положения блоков с помощью позиционирования.

6.15. Использование плавающих элементов

"Плавающим элементом" называется элемент, обтекаемый с разных сторон текстом или другими объектами веб-страницы. Параметр float, добавленный к блоку, определяет по какой стороне будет выравниваться блок, при этом остальные элементы будут обтекать его с других сторон.

Рассмотрим вариант, когда левый блок имеет определенный размер, а ширина правого блока устанавливается автоматически, исходя из ширины окна браузера. При этом ширина левого блока может задаваться в пикселах или процентах. В табл. 6.3 приведены основные стилевые параметры для формирования двух блоков.

Таблица 6.3 – Примеры резиновой блочной верстки с заданной шириной левого блока

Для левого блока шириной 20 %	**Для левого блока шириной 200px**

Блок 1 float: left width: 20 %	Блок 2 margin-left: 21 %	Блок 1 float: left width: 200px	Блок 2 margin-left: 210px

Для блока 1 требуется всего два параметра: float – располагает блок 2 рядом с блоком 1 по горизонтали и width – устанавливает ширину блока 1. Блок 2 будет занимать все оставшееся место, поэтому для него атрибут width не требуется. Блок 2 характеризуется лишь одним параметром – margin-left, который смещает левый край блока 2 на ширину блока 1, плюс задает отступ между блоками. Поэтому значение этого свойства в таблице 6.3 равно 21 %, где 20 % – это ширина блока 1, а 1 % – расстояние между блоками. Чтобы задать ширину одного из блоков в пикселах, можно использовать тот же код, изменив только единицы измерения. Для формирования блока заданной ширины справа, а не слева, код незначительно модифицируется (табл. 6.4).

Таблица 6.4 – Примеры резиновой блочной верстки с заданной шириной правого блока

Для левого блока шириной 20%		**Для левого блока шириной 200px**	
Блок 1 float: right width: 20 %	Блок 2 margin-right: 21 %	Блок 1 float: right width: 200px	Блок 2 margin-right: 210px

Расположение блоков в коде остается прежним, но значение атрибута float изменяется на right, а параметр отступа – на margin-right.

6.16. Применение позиционирования

При формировании двухколонного макета левая или правая колонка устанавливается в определенную позицию с помощью абсолютного позиционирования, а соседняя колонка освобождает для нее место за счет использования отступов. Ниже представлен пример кода, создающего две колонки, при этом ширина левой колонки равна 200 пикселов, а ее положение определяется по отношению к левому верхнему углу окна браузера.

#leftcol { / Левая колонка */*
position: absolute; / Абсолютное позиционирование */*
width: 200px; / Ширина слоя */*
left: 0; / Положение от левого края окна */*
top: 20px; / Положение от верхнего края окна */*
background: #fc0; / Цвет фона левой колонки */ }*
#rightcol { / Правая колонка */*
margin-left: 210px; / Отступ слева */*
background: #ccc; / Цвет фона правой колонки */ }*

Значение absolute свойства position позволяет задавать положение блока относительно края окна браузера независимо от наличия и местоположения других блоков. Сами координаты устанавливаются с помощью параметров left, top, right и bottom, которые соответственно определяют расстояние от левого верхнего, правого и нижнего края окна. Чтобы левая и правая колонка не накладывались друг на друга, следует добавить отступ слева (margin-left) для блока rightcol, как показано в данном примере. Значение этого отступа включает расстояние от левого края (параметр left) и ширину левой колонки (width) плюс расстояние между колонками.

6.17. Использование блочной верстки

Ниже представлен пример использования блочной верстки для создания шапки сайта (рис. 6.7).

Рис. 6.7. Шапка сайта

Для реализации этого примера можно использовать такой код:

* *
* Логотип *
* Контактные реквизиты *
* *

Далее для главного тега <DIV id="header"> можно установить рамку

и залить фон вложенных тегов <DIV> разными цветами (рис. 6.8):

```
<style type="text/css">
div#header { border: green 2px dashed; }
div#logo { background: yellow; }
div#contact { background: blue; }
</style>
```

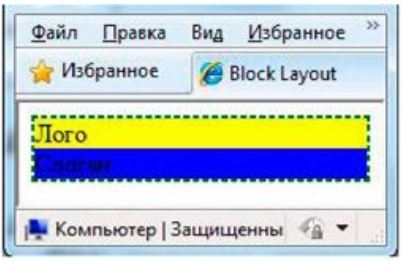


Рис. 6.8. Установка рамки и заливка фона

Затем следует задать размеры внутренних блоков (рис. 6.9):

```
div#logo { background: yellow;
width: 100px;
height: 100px; }
div#contact { background: blue;
width: 100px;
height: 100px; }
```

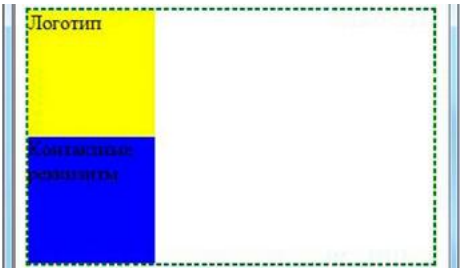


Рис. 6.9. Задание размеров блоков

Следующей задачей будет размещение блоков на одной линии (в одной строке). Для этого можно воспользоваться свойством float, которое задает обтекание текста слева, справа или вокруг того блока, к которому

применяется это свойство. В данном примере нужно задать обтекание для первого блока, чтобы второй располагался справа от него. Для этого нужно для свойства float установить значение left (значение указывает положение элемента, который будут обтекать соседние элементы). Но обтекать блок будут только встроенные элементы, а блочные (в данном случае <DIV>) будут располагаться под ним (с меньшим значением координаты свойства Z-index). Поэтому лучше сразу сместить второй блок к правому краю родительского. Это можно сделать, установив для левого внешнего отступа второго блока (margin-left) значение auto, при котором отступ будет "занимать" всю доступную ширину (рис. 6.10):

```
div#logo { background-color: yellow;
width: 100px;
height: 100px;
float: left; }
div#contact { background-color: blue;
width: 100px;
height: 100px;
margin-left: auto; }
```

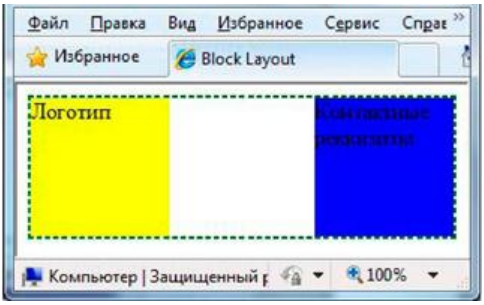


Рис. 6.10. Установка обтекания и выравнивание

Последним этапом работы будет смещение левого блока вверх и влево, а правого вверх и вправо. Для этого блокам нужно для свойства позиционирования (position) установить значение relative и указать значения смещений: left, top, right. (рис. 6.11):

```
div#logo { background-color: yellow;
```

width: 100px;
height: 100px;
float: left;
position: relative;
top: –5px;
left: –5px; }
div#contact { background-color: blue;
width: 100px;
height: 100px;
margin-left: auto;
position: relative;
top: –5px;
right: –5px; }

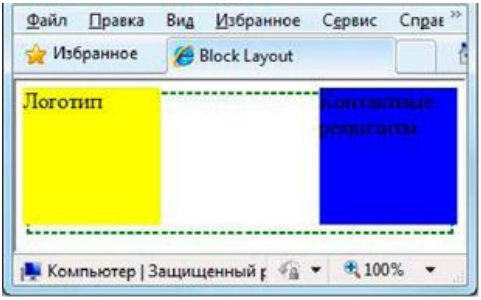


Рис. 6.11. Установка относительного позиционирования и значений смещений

Практические задания

1. Сверстать шаблон страниц сайта, используя приемы блочной верстки, согласно варианту задания.
2. При создании шаблона строго выдержать принцип разделения структуры (применять HTML) и представления (применять только CSS).
3. Использовать "резиновый" макет (каркас) шаблона.
4. Задать минимальную ширину каркаса. Обосновать свой выбор.
5. Определить какие блоки шаблона будут резиновыми, а какие фиксированными. Обосновать свой выбор.

6. Все стили разместить во внешнем(их) файле(ах).

7. Предусмотреть наборы стилей как минимум для двух устройств (screen и printer).

8. Проверить шаблон на соответствие стандартам W3C и на совместимость с наиболее популярными браузерами.

9. На основании шаблона создать сайт, состоящий минимум из 5 страниц.

10. Тематика сайта – произвольная.

Варианты заданий

Вариант 1

<table>
<tr><td>Логотип</td><td></td><td>Контактные реквизиты</td></tr>
<tr><td colspan="2">Горизонтальное меню</td><td>Поиск</td></tr>
<tr><td>Вертикальное меню</td><td colspan="2">Рисунок по теме сайта</td></tr>
<tr><td><table><tr><td>Новости
-
-
-</td></tr><tr><td>Статьи
-
-
-</td></tr></table></td><td colspan="2">Основное содержимое</td></tr>
<tr><td colspan="3">©</td></tr>
</table>

Вариант 2

<table>
<tr><td>Логотип</td><td>Контактные реквизиты</td></tr>
</table>

Вертикальное меню Счетчики	Основное содержимое	Реклама Google
©		

Вариант 3

Название компании Миссия компании				
Главное горизонтальное меню:				
Рисунок 1	Рисунок 2	Рисунок 3	Рисунок 4	Рисунок 5
Изображение по теме текущей страницы		Основное содержимое		
Последние новости - - -				
Пункт 1	Пункт 2	Пункт 3	Пункт 4	Пункт 5
©				

Вариант 4

Логотип		Счетчики (HotLog, LiveInternet)
Горизонтальное меню		
Основное содержимое		Вертикальное подменю, пункты которого зависят от выбранного

	пункта горизонтального меню
Рекламные баннеры	
©	

Вариант 5

Логотип	Поиск	
Вертикальное главное меню	Рисунок текущего пункта страницы	Контекстная реклама
	Полное название текущей страницы	Новости
	Основное содержимое	- новость 1 - новость 2 - новость 3
Контактная информация	Логотипы партнеров	
Горизонтальное главное меню		
©		

Вариант 6

Логотип	Главное вертикальное меню	Рисунок по теме сайта
Расширенное название текущей страницы Аннотация (резюме) текущей страницы		
Вертикальное подменю	Новости & События	
	- новость 1 - новость 2 - новость 3	- событие 1 - событие 2 - событие 3

<table>
<tr><td rowspan="3">Подробные
контактные
реквизиты</td><td>Основное содержимое</td></tr>
<tr><td>Главное горизонтальное меню</td></tr>
<tr><td>©</td></tr>
</table>

Вариант 7

<table>
<tr><td>Логотип</td><td>Основные страницы в
виде пиктограмм</td></tr>
<tr><td>Вертикальное меню</td><td>Изображение по теме сайта</td></tr>
<tr><td>Развернутое название
текущей страницы</td><td>Аннотация (резюме)
текущей страницы</td></tr>
<tr><td rowspan="3">События</td><td>Основное содержимое</td></tr>
<tr><td>Главное горизонтальное меню</td></tr>
<tr><td>©</td></tr>
</table>

Вариант 8

<table>
<tr><td>Контактные
реквизиты</td><td>Слоган (девиз)
компании</td><td>Логотип</td></tr>
<tr><td>Поиск</td><td colspan="2">Горизонтальное меню</td></tr>
<tr><td colspan="2">Рисунок по теме сайта</td><td>Вертикальное
подменю</td></tr>
</table>

<table>
<tr><td rowspan="2">Основное содержимое</td><td>Новости
-
-
-</td></tr>
<tr><td>Статьи
-
-
-</td></tr>
<tr><td>Главное горизонтальное меню
©</td><td></td></tr>
</table>

Вариант 9

<table>
<tr><td rowspan="2">Изображение
по тематике сайта,
включающее в себя
логотип</td><td colspan="2">Главное горизонтальное меню</td></tr>
<tr><td>Развернутое
название
текущей страницы</td><td>Изображение
по теме
текущей страницы</td></tr>
<tr><td colspan="2" rowspan="2">Основное содержимое</td><td>Новости</td></tr>
<tr><td>- новость 1
- новость 2
- новость 3</td></tr>
<tr><td>Контактная
информация</td><td colspan="2">Поиск по сайту</td></tr>
<tr><td colspan="3">Главное горизонтальное меню</td></tr>
<tr><td colspan="3">©</td></tr>
</table>

Вариант 10

<table>
<tr><td>Горизонтальное меню в виде картинок
(при наведении курсора на пункт он увеличивается на 15%)</td></tr>
<tr><td>Изображение на всю ширину по тематике сайта,
включающее в себя логотип</td></tr>
<tr><td>Основное содержимое</td></tr>
<tr><td>Реклама</td></tr>
</table>

©

Контрольные вопросы

1. Что такое блочная модель сайта?
2. В чем разница между полями и отступами?
3. С помощью какого свойства можно создать блочный элемент?
4. С помощью какого свойства можно создать строчный элемент?
5. Какие свойства позволяют регулировать наложение и порядок слоев?
6. С помощью какого свойства можно задать плавающий элемент?
7. Перечислите виды позиционирования?
8. Проведите сравнительный анализ табличной и "резиновой" верстки. Укажите их преимущества и недостатки.

7. ОСНОВЫ РАБОТЫ С JAVASCRIPT

7.1. Основные сведения

JavaScript – это язык программирования, позволяющий сделать веб-страницу интерактивной, то есть реагирующей на действия пользователя.

Последовательность инструкций (называемая программой, скриптом или сценарием) выполняется интерпретатором, встроенным в обычный веб-браузер, т.е. код программы внедряется в HTML-документ и выполняется на стороне клиента. Для выполнения программы не нужно перезагружать веб-страницу. Все программы выполняются в результате возникновения события. Например, перед отправкой данных формы можно проверить их на допустимые значения и если значения не соответствуют ожидаемым, запретить отправку данных.

7.2. Создание первой программы на JavaScript

При изучении языков программирования принято начинать с программы, выводящей надпись "Hello, world". Ниже представлен программный код такой программы, написанный на JavaScript.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Первая программа</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript"> document.write("Hello, world"); </script>
<noscript>
<p>Ваш веб-браузер не поддерживает JavaScript</p>
</noscript>
</body>
</html>
```

Код необходимо набрать в редакторе Notepad++, сохранить в формате HTML, например, под именем test.html. Программа внедряется в HTML-документ с помощью парного тега <script>. В качестве значения параметра type указывается MIME-тип text/javascript. Кроме того, может быть указан параметр language, который задает название языка программирования (в данном примере – JavaScript). Данный параметр использовался в ранних версиях HTML, а в настоящее время указывается только для совместимости одновременно с параметром type:

<script type="text/javascript" language="JavaScript">

Если веб-браузер не поддерживает JavaScript или выполнение скриптов запрещено в настройках веб-браузера, то будет выведен текст между тегами <noscript> и </noscript>.

Строка, содержащая инструкцию отобразить надпись "Hello, world" в окне веб-браузера, называется выражением. Например:

document.write("Hello, world");

Каждое выражение в JavaScript заканчивается точкой с запятой. В принципе, это требование не является обязательным. Тем не менее, рекомендуется ставить точку с запятой в конце каждого выражения. Это позволит избежать множества ошибок в дальшейшем.

При открытии файла test.html в веб-браузере возможны следующие варианты:

- в окне веб-браузера отображена надпись "Hello, world", следовательно, программа работает корректно;
- отобразилась надпись "Ваш веб-браузер не поддерживает JavaScript" и веб-браузер задает вопрос "Запустить скрипты?", следовательно, в настройках веб-браузера выбрана опция "Подтверждать запуск скриптов". В этом случае можно либо выбрать оцию "Разрешить запуск сприптов", либо каждый раз отвечать "Да" на этот вопрос;
- отобразилась надпись "Ваш веб-браузер не поддерживает JavaScript" и веб-браузер не задает никаких вопросов, следовательно, в настройках веб-браузера выбрана опция "Запретить запуск скриптов". В этом случае необходимо выбрать опцию "Разрешить запуск сприптов";

- в окне веб-браузера нет никаких надписей, следовательно, допущена ошибка в коде программы. В JavaScript регистр имеет важное значение: строчные и прописные буквы считаются разными. Более того, каждый символ (буквы, знаки препинания и т.п.) имеет значение.

Таким образом, при использовании JavaScript возникает несколько проблем:

1. Любой пользователь может отключить запуск скриптов в настройках веб-браузера.
2. Разные веб-браузеры могут по-разному выполнять код программы.

7.3. Комментарии в JavaScript

Все, что расположено после "//" до конца строки, в JavaScript считается однострочным комментарием. Однострочный комментарий можно записать после выражения. Например:

document.write("Hello, world"); // Однострочный комментарий

Кроме того, существует многострочный комментарий. Он начинается с символов /* и заканчивается символами */.

7.4. Вывод результатов работы программы и ввод данных

Для вывода результатов работы программы или значений переменных, а также для ввода данных существуют специальные встроенные диалоговые окна.

Окно с сообщением и кнопкой OK

Метод alert() отображает диалоговое окно с сообщением и кнопкой OK. Ниже представлен программный код, демонстрирующий вывод приветствия с помощью метода alert().

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Метод alert()</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
```

```
</head>
<body>
<script type="text/javascript"> window.alert("Hello, world"); </script>
</body>
</html>
```

Сообщение можно разбить на строки с помощью последовательности символов \n. Например:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR__
<html>
<head><title>Разбиение сообщения на строки</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript">
window.alert("Строка1\nСтрока2\n\nСтрока4");
</script>
</body>
</html>
```

Окно с сообщением и кнопками OK и Cancel

Метод confirm() отображает диалоговое окно с сообщением и двумя кнопками OK и Cancel. Он возвращает значение true, если нажата кнопка OK, и false – если Cancel. Ниже представлен пример использования в коде метода confirm().

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head><title>Метод confirm()</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript">
```

```
if (window.confirm("Нажмите на одну из кнопок")) {
window.alert("Нажата кнопка OK"); }
else { window.alert("Нажата кнопка Cancel"); }
</script>
</body>
</html>
```

Окно с полем ввода и кнопками OK и Cancel

Метод prompt() отображает диалоговое окно с сообщением, полем ввода и двумя кнопками OK и Cancel. Он возвращает введенное значение, если нажата кнопка OK, или специальное значение null, если нажата кнопка Cancel. Ниже представлен пример использования в коде метода prompt().

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head><title>Метод prompt()</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript">
var n = window.prompt("Введите ваше имя", "Это значение по умолчанию");
if (n==null) { document.write("Вы нажали Cancel"); }
else {document.write("Привет " + n); } </script>
</body>
</html>
```

7.5. Переменные

Переменные – это участки памяти, используемые программой для хранения данных. Каждая переменная должна иметь уникальное имя в программе, состоящее из латинских букв, цифр и знаков подчеркивания (например, *x*, *frame1*, *_name*). Первым символом может быть либо буква, либо знак подчеркивания (например, *strName*, *_y1*). В названии переменной может также присутствовать символ $ (например, *salary$*). Имена

переменных не должны совпадать с зарезервированными ключевыми словами языка JavaScript (например, *frame* – это имя некорректное, так как является ключевым словом).

При указании имени переменной важно учитывать регистр. Например, *strName* и *strname* – это две разные переменные.

В программе переменные объявляются с помощью ключевого слова var. Например: *var strName;*

Можно объявить сразу несколько переменных в одной строке, указав их через запятую. Например: *var x, strName, y1, _name, frame1;*

7.6. Типы данных и инициализация переменных

В JavaScript переменные могут содержать следующие типы данных:

- number – целые числа или числа с плавающей точкой (дробные);
- string – строки;
- boolean – логический тип данных. Может содержать значения true (истина) или false (ложь);
- function – функции. В языке JavaScript ссылку на функцию можно присвоить какой-либо переменной. Для этого название функции указывается без круглых скобок. Кроме того, функции имеют свойства и методы;
- object – массивы, объекты, а также переменная со значением null.

При инициализации переменной JavaScript автоматически относит переменную к одному из типов данных. При этом под инициализацией переменных понимают операцию присваивания переменной начального значения. Значение переменной присваивается с помощью оператора =. Например:

```
Number1 = 7; // Переменной Number1 присвоено значение 7
Number2 = 7.8; // Переменной Number2 присвоено значение с плавающей точкой
String1 = "Строка"; // Переменной String1 присвоено значение Строка
String2 = 'Строка'; // Переменной String2 также присвоено значение Строка
Boolean1 = true; // Переменной Boolean1 присвоено логическое значение true
Str1 = null; // Переменная Str1 не содержит данных
```

Переменной может быть присвоено начальное значение сразу при ее объявлении. Можно задать начальные значения сразу нескольким пере-

менным. Например:

```
var str1 = "Строка";
var str2 = "Строка", Number1 = 7;
```

Если в программе обратиться к переменной, которая не объявлена, то возникнет критическая ошибка. Если переменная объявлена, но ей не присвоено начальное значение, то значение полагается равным undefined.

Оператор typeof возвращает строку, описывающую тип данных переменной. Ниже представлен пример использования в коде оператора typeof.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Типы данных</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript">
var Number1 = 7;
var Number2 = 7.8;
var String1 = "Строка";
var String2 = 'Строка';
var Boolean1 = true;
var Str1 = null, Str2;
document.write("Number1 – " + typeof (Number1) + "<br>");
document.write("Number2 – " + typeof (Number2) + "<br>");
document.write("String1 – " + typeof (String1) + "<br>");
document.write("String2 – " + typeof (String2) + "<br>");
// Скобки можно не указывать
document.write("Boolean1 – " + typeof Boolean1 + "<br>");
document.write("Str1 – " + typeof Str1 + "<br>");
document.write("Str2 – " + typeof Str2);
</script>
```

</body>
</html>

7.7. Операторы JavaScript

Операторы позволяют выполнить действия с данными. Например, операторы присваивания предназначены для сохранения данных в переменной, математические операторы позволяют произвести арифметические вычисления, а оператор конкатенации строк используется для соединения двух строк в одну. Операторы берут одно или два значения, представляющие собой переменную, константу или другое выражение, содержащее операторы или функции, и возвращают одно значение, определяемое по исходным данным. В JavaScript доступны такие виды операторов.

Математические операторы

- + – сложение. Например: *Z = X + Y*;
- – – вычитание. Например: *Z = X – Y*;
- * – умножение. Например: *Z = X * Y*;
- / – деление. Например: *Z = X / Y*;
- % – деление по модулю. Например: *Z = X % Y*;
- ++ – оператор инкремента. Увеличивает значение переменной на 1. Например: *Z++; // Эквивалентно Z = Z + 1*;
- -- – оператор декремента. Уменьшает значение переменной на 1: *Z--; // Эквивалентно Z = Z – 1*;

Операторы инкремента и декремента могут использоваться в постфиксной или префиксной формах. Например:

```
Z++; Z--; // Постфиксная форма
++Z; --Z; // Префиксная форма
```

При постфиксной форме (Z++) возвращается значение, которое переменная имела перед операцией, а при префиксной форме (++Z) – вначале производится операция и только потом возвращается значение. Ниже представлен пример кода, демонстрирующего отличия между этими формами.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

```
<html>
<head>
<title>Постфиксная и префиксная форма</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript">
var X, Y;
X = 5;
Y = X++; // Y = 5, X = 6
var msg;
msg = "<b>Постфиксная форма (Y = X++;):" + "</b><br> Y = ";
msg += Y + "<br>X = " + X + "<br><br>";
X = 5;
Y = ++X; // Y = 6, X = 6
msg += "<b>Префиксная форма (Y = ++X;):" + "</b><br> Y = ";
msg += Y + "<br>X = " + X;
document.write(msg);
</script>
</body>
</html>
```

Операторы присваивания

- = присваивает переменной значение. Например: *Z = 5*;
- += увеличивает значение переменной на указанную величину. Например: *Z += 5; // Эквивалентно Z = Z + 5*;
- –= уменьшает значение переменной на указанную величину. Например: *Z –= 5; // Эквивалентно Z = Z – 5*;
- *= умножает значение переменной на указанную величину. Например: *Z *= 5; // Эквивалентно Z = Z * 5*;
- /= делит значение переменной на указанную величину. Например: *Z /= 5; // Эквивалентно Z = Z / 5*;
- %= делит значение переменной на указанную величину и возвра-

щает остаток. Например: *Z %= 5; // Эквивалентно Z = Z % 5*;

Двоичные операторы

Двоичные операторы выполняют поразрядные действия с двоичным представлением целых чисел.

- ~ – двоичная инверсия. Например: *Z = ~X*;
- & – двоичное И. Например: *Z = X & Y*;
- | – двоичное ИЛИ. Например: *Z = X | Y*;
- ^ – двоичное исключающее ИЛИ. Например: *Z = X ^ Y*;
- << – сдвиг влево, т.е. сдвиг влево на один или более разрядов с заполнением младших разрядов нулями. Например: *Z = X << Y*;
- >> – сдвиг вправо, т.е. сдвиг вправо на один или более разрядов с заполнением старших разрядов содержимым самого старшего разряда. Например: *Z = X >> Y*;
- >>> – сдвиг вправо без учета знака, т.е. сдвиг вправо на один или более разрядов с заполнением старших разрядов нулями. Например: *Z = X >>> Y*.

Оператор обработки строк

- \+ – оператор конкатенации строк. Например:

```
var Str = "Строка1" + "Строка2";
// Переменная Str будет содержать значение "Строка1Строка2"
```

Часто необходимо сформировать строку, состоящую из имени переменной и ее значения. Если написать, например:

```
var X = "Строка1";
var Z = "Значение равно X";
```

то переменная Z будет содержать значение "Значение равно X".

Но если несколько изменить эту запись, например, так:

```
var X = "Строка1";
var Z = "Значение равно " + X;
```

то переменная Z будет содержать значение "Значение равно Строка1".

Ниже представлен программный код, демонстрирующий вывод значения переменной в диалоговом окне.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

```
<html>
<head>
<title>Вывод значения переменной</title>
<meta http-equiv="Content-Type"content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript">
var X = "Строка1";
window.alert("Переменная X содержит значение 'X' ");
// Выведет "Переменная X содержит значение 'X' "
window.alert("Переменная X содержит значение "+ X);
// Выведет "Переменная X содержит значение 'Строка1'"
</script>
</body>
</html>
```

7.8. Приоритет выполнения операторов

Ниже представлен список операторов в порядке убывания приоритета:

1. !, ~, ++, -- – отрицание, двоичная инверсия, инкремент, декремент.
2. *, /, % – умножение, деление, остаток от деления.
3. +, - – сложение и вычитание.
4. <<, >>, >>> – двоичные сдвиги.
5. & – двоичное И.
6. ^ – двоичное исключающее ИЛИ.
7. | – двоичное ИЛИ.
8. =, +=, -=, *=, /=, %= – присваивание.

Например, для выражения:

```
X = 5 + 10 * 3 / 2;
```

последовательность выполнения операторов будет следующей:

1. Оператор умножения, т.е. число 10 будет умножено на 3, так как приоритет операции умножения выше приоритета операции сложения.
2. Оператор деления. Полученное на этапе 1 значение будет поделено

на 2, так как приоритет операции деления равен операции умножения, но выше операции сложения. При равных приоритетах операции выполняются слева направо.

3. Оператор сложения. К полученному на этапе 2 значению будет прибавлено число 5, так как приоритет операций сложения ниже приоритетов операций умножения и деления.

4. Оператор присваивания. Значение, полученное на этапе 3, будет присвоено переменной X, т.к. оператор присваивания имеет наименьший приоритет среди операций этого выражения.

Однако с помощью скобок можно изменить последовательность вычисления выражения. Следующее выражение будет вычислено в таком порядке:

*X = (5 + 10) * 3 / 2;*

1. К числу 5 будет прибавлено 10.
2. Полученное значение будет умножено на 3.
3. Полученное значение будет поделено на 2.
4. Значение будет присвоено переменной X.

7.9. Преобразование типов данных

Иногда при создании программ возникает необходимость производить операции с переменными разных типов данных. В этом случае осуществляется автоматическое преобразование данных. Например, если к числу прибавить строку, интерпретатор столкнется с несовместимостью типов данных и попытается преобразовать переменные к одному типу данных, а затем выполнить операцию. Например:

```
var Str = "5";
var Number1 = 3;
var Str2 = Number1 + Str; // Переменная содержит строку "35"
var Str3 = Str + Number1; // Переменная содержит строку "53"
```

В этом примере переменная Number1, имеющая тип number (число), будет преобразована к типу string (строка), а затем будет произведена операция конкатенации строк.

Если из числа вычесть строку, число умножить на строку или число

разделить на строку, интерпретатор попытается преобразовать строку в число, а затем вычислить выражение. Причем не важно, в какой последовательности будут указаны число и строка. Например:

```
var Number1 = 15;
var Str = "5";
var Str2 = Number1 – Str; // Переменная содержит число 10
var Str3 = Number1 * Str; // Переменная содержит число 75
var Str4 = Number1 / Str; // Переменная содержит число 3
var Str5 = Str * Number1; // Переменная все равно содержит число 75
```

Но если строка будет состоять из букв, интерпретатор не сможет преобразовать строку в число и присвоит переменной значение NaN (Not a Number, не число). Например:

```
var Number1 = 15;
var Str = "Строка";
var Str2 = Number1 – Str; // Переменная содержит значение NaN
```

Таким образом, интерпретатор осуществляет преобразование типов данных автоматически. Однако, в таком случае можно получить результат, отличающийся от ожидаемого. Поэтому лучше оперировать переменными одного типа, а если необходимо осуществить преобразования типов, то делать это вручную.

Для преобразования типов данных можно использовать следующие встроенные функции JavaScript:

- parseInt(<Строка>, [<Основание>]) преобразует строку в целое число. Строка считается заданной в системе счисления, указанной вторым необязательным параметром. Если основание не указано, то по умолчанию используется десятичная система. Если строка не может быть преобразована в число, возвращается значение NaN (Not a Number, не число). Например:

```
var Number1 = 15;
var Str = "5";
var Str5 = "FF";
var Str2 = Number1 – parseInt(Str); // Переменная содержит число 10
```

```
var Str3 = Number1 – parseInt(Str5, 16); // Переменная содержит число –240
var Str4 = Number1 + parseInt(Str); // Переменная содержит число 20
```

- parseFloat(<Строка>) преобразует строку в число с плавающей точкой. Например:

```
var Str = "5.2";
var Str2 = parseFloat(Str); // Переменная содержит число 5.2
```

- eval(<Строка>) вычисляет выражение в строке, как обычное выражение JavaScript. Например:

```
var Str = "3 + 5";
var Str2 = eval(Str); // Переменная содержит число 8
```

Ниже представлен пример использования в коде преобразования типов данных. Программа суммирует два числа, введенных пользователем в поля двух диалоговых окон.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Вычисление суммы двух чисел</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript">
var Str1, Str2, Sum1, Sum2, msg;
Str1 = window.prompt("Вычисление суммы двух чисел\nВведите число 1", "");
if (Str1==null) { document.write("Вы нажали Отмена"); }
else { Str2=window.prompt("Вычисление суммы двух чисел\nВведите число 2", "");
if (Str2==null) { document.write("Вы нажали Отмена"); }
else { Sum1 = Str1 + Str2;
msg = "До преобразования типов:<br>Значение суммы чисел ";
msg += Str1 + " и " + Str2 + " равно ";
msg += Sum1 + "<br><br>";
Sum2 = parseInt(Str1) + parseInt(Str2);
```

```
msg += "После преобразования типов:<br>";
msg += "Значение суммы чисел " + Str1 + " и ";
msg += Str2 + " равно " + Sum2;
document.write(msg); } }
</script>
</body>
</html>
```

Если в обоих диалоговых окнах набрать число 5, то в окне веб-браузера отобразится следующий текст:

До преобразования типов:
Значение суммы чисел 5 и 5 равно 55
После преобразования типов:
Значение суммы чисел 5 и 5 равно 10

Таким образом, диалоговые окна возвращают в качестве типа значения строку. Чтобы получить сумму двух чисел, указанных в полях диалоговых окон, нужно обязательно осуществить преобразование типов данных.

7.10. Специальные символы

Специальные символы – это комбинации знаков, обозначающих служебные или непечатаемые символы, которые невозможно вставить обычным способом.

Именно с помощью специального символа \n (перевод строки) мы разбиваем сообщение в диалоговом окне на строки (см. пример кода ниже).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Специальные символы</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript">
```

window.alert("Строка1\nСтрока2\n\nСтрока4"); </script>
</body>
</html>

В JavaScript доступны такие специальные символы:

- \n – перевод строки;
- \r – возврат каретки;
- \f – перевод страницы;
- \t – знак табуляции;
- \' – апостроф;
- \" – кавычка;
- \\ – обратная косая черта.

7.11. Массивы

Массив – это нумерованный набор переменных. Переменная в массиве называется элементом массива, а ее позиция в массиве задается индексом. Нумерация элементов массива начинается с 0, а не с 1. Это следует помнить. Общее количество элементов в массиве называется размером массива. При инициализации массива переменные указываются через запятую в квадратных скобках. Например: *Mass1 = [1, 2, 3, 4];*

Получить значение элемента массива можно, указав его индекс в квадратных скобках. Например:

Str = Mass1[0]; // Переменной Str будет присвоено значение 1

Ниже представлен пример кода, демонстрирующего создание массива и вывод значения элемента массива в окне веб-браузера.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Массивы</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>

```
<script type="text/javascript">
var Mass1, Mass2;
Mass1 = [1, 2, 3, 4];
Mass2 = ["Январь", "Февраль", "Март", "Апрель", "Май", "Июнь",
"Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"];
document.write(Mass1[1] + " и " + Mass2[2]); </script>
</body>
</html>
```

В случае необходимости можно добавить новый элемент массива или изменить значение существующего. Например:

```
Mass1[5] = 6;
Mass1[0] = 0;
```

В этом примере было создано два элемента массива и изменено значение существующего. Первый элемент с индексом 5 создан пользователем, а элемент с индексом 4 будет создан автоматически, ему будет присвоено значение undefined (неопределен), так как наш массив состоял только из 4 элементов, и последний определенный элемент имел индекс 3.

Любому элементу массива можно присвоить другой массив. Например:

```
Mass1[0] = [1, 2, 3, 4];
```

В этом случае получить значение массива можно, указав два индекса. Например:

```
Str = Mass1[0][2]; // Переменной Str будет присвоено значение 3
```

Следует учитывать, что операция присваивания сохраняет в переменной ссылку на массив, а не все его значения (см. пример кода ниже).

```
var Mass1, Mass2;
Mass1 = [1, 2, 3, 4];
Mass2 = Mass1; // Присваивается ссылка на массив!!!
Mass2[0] = "Новое значение";
document.write(Mass1.join(", ") + "<br>");
document.write(Mass2.join(", "));
```

В этом примере изменение Mass2 затронет Mass1, и будет получен следующий результат:

Новое значение, 2, 3, 4

Новое значение, 2, 3, 4

Чтобы сделать копию массива можно, например, воспользоваться методом slice(), который возвращает срез массива (см. пример кода ниже).

```
var Mass1, Mass2;
Mass1 = [1, 2, 3, 4];
Mass2 = Mass1.slice(0);
Mass2[0] = "Новое значение";
document.write(Mass1.join(", ") + "<br>");
document.write(Mass2.join(", "));
```

Результат:

1, 2, 3, 4

Новое значение, 2, 3, 4

При использовании многомерных массивов метод slice() создает "поверхностную" копию, а не полную. Например:

```
var Mass1, Mass2;
Mass1 = [[0, 1], 2, 3, 4];
Mass2 = Mass1.slice(0);
Mass2[0][0] = "Новое значение1";
Mass2[1] = "Новое значение2";
```

В результате массивы будут выглядеть так:

Mass1 = [["Новое значение1", 1], 2, 3, 4];

Mass2 = [["Новое значение1", 1], "Новое значение2", 3, 4];

В данном примере изменение вложенного массива в Mass2 привело к одновременному изменению значения в Mass1, т.е. оба массива содержат ссылку на один и тот же вложенный массив.

7.12. Функции. Разделение программы на фрагменты

Функция – это фрагмент кода JavaScript, который можно вызвать из любого места программы. Функция описывается с помощью ключевого

слова function по следующей схеме:

function <Имя функции> ([<Параметры>]) {

<Тело функции>

[return <Значение>] }

Функция должна иметь уникальное имя. Для имен функций действуют такие же правила, как и для имен переменных.

После имени функции в круглых скобках можно указать один или несколько параметров через запятую. Параметров может вообще не быть. В этом случае указываются только круглые скобки.

Между фигурными скобками располагаются выражения JavaScript. Кроме того, функция может возвращать значение вместо вызова функции. Возвращаемое значение задается с помощью ключевого слова return.

Пример функции без параметров:

function f_Alert_OK() {

window.alert("Сообщение при удачно выполненной операции"); }

Пример функции с параметром: *function f_Alert(msg) {window.alert(msg);}*

Пример функции с параметрами, возвращающей сумму двух переменных: *function f_Sum(x, y) { var z = x + y; return z; }*

В качестве возвращаемого значения в конструкции return можно указывать не только имя переменной, но и выражение. Например: *function f_Sum(x, y) { return (x + y); }*

В программе функции можно вызвать, например, таким образом:

f_Alert_OK();

f_Alert("Сообщение");

Var1 = f_Sum(5, 2); // Переменной Var1 будет присвоено значение 7

Выражения, указанные после return <значение>;, никогда не будут выполнены. Например:

function f_Sum(x, y) { return (x + y);

window.alert("Сообщение"); // Это выражение никогда не будет выполнено }

Имя переменной, передающей значение функции, может не совпадать с именем переменной внутри функции. Например:

function f_Sum(x, y) { return (x + y); }

var Var3, Var1 = 5;
var Var2 = 2;
Var3 = f_Sum (Var1, Var2);

Ссылку на функцию можно сохранить в какой-либо переменной. Для этого название функции указывается без круглых скобок. Например:

function test() {
window.alert("Это функция test()"); }
var x;
x = test; // Присваиваем ссылку на функцию
x(); // Вызываем функцию test() через переменную x

Кроме того, функция может вообще не иметь названия. Такая функция называется анонимной. Это особый вид функций, которые объявляются в месте использования и не получают уникального идентификатора для доступа к ним. Обычно при создании они либо вызываются напрямую, либо ссылка на функцию присваивается переменной, с помощью которой затем можно косвенно вызывать данную функцию. В этом случае ссылку на анонимную функцию сохраняют в переменной. Например:

var x = function() { // Присваиваем ссылку на анонимную функцию
window.alert("Сообщение"); };
x(); // Вызываем анонимную функцию через переменную x

Ссылку на вложенную функцию можно вернуть в качестве значения в инструкции return. Чтобы вызвать вложенную функцию круглые скобки указываются два раза. Например:

var x = function() { // Присваиваем ссылку на анонимную функцию
return function() { // Возвращаем ссылку на вложенную функцию
window.alert("Это вложенная функция"); }; };
x()(); // Вызываем вложенную функцию через переменную x

Обычно функции принято располагать в разделе HEAD HTML-документа или в отдельном файле с расширением js (создать файл с таким расширением можно в редакторе Notepad++). Хотя функции могут располагаться и в разделе BODY.

Пример функции, расположенной в разделе HEAD:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Функции</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<script type="text/javascript">
function f_Sum(x, y) { return (x + y); }
</script>
</head>
<body>
<script type="text/javascript">
var Var3, Var1 = 5, Var2 = 3;
Var3 = f_Sum(Var1, Var2);
document.write(Var3);
</script>
</body>
</html>
```

Пример функции, вынесенной в отдельный файл script.js:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Функции</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<script type="text/javascript">
var Var3, Var1 = 5, Var2 = 3;
Var3 = f_Sum(Var1, Var2);
```

```
document.write(Var3);
</script>
</body>
</html>
```

Содержимое файла script.js:

```
function f_Sum(x, y) { return (x + y); }
```

7.13. Рекурсия

Рекурсия – это возможность функции вызывать саму себя. Однако если не предусмотреть условие выхода, происходит зацикливание. Ниже представлен пример программы, вычисляющей факториал.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Вычисление факториала</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<script type="text/javascript">
function f_Factorial(x) { if (x == 0 || x == 1) return 1;
else return (x * f_Factorial(x - 1)); }
</script>
</head>
<body>
<script type="text/javascript">
var z;
z = window.prompt("Вычисление факториала\nВведите число", "");
if (z==null) { document.write("Вы нажали Отмена"); }
else { document.write("Факториал числа " + z + " = ");
document.write(f_Factorial(parseInt(z))); }
</script>
</body>
</html>
```

7.14. Глобальные и локальные переменные

Глобальные переменные – это переменные, которые объявлены вне функции и видны в любой части программы, включая функции.

Локальные переменные – это переменные, которые объявлены внутри функции и видны только внутри тела функции. Если имя локальной переменной совпадает с именем глобальной переменной, то все операции внутри функции осуществляются с локальной переменной, а значение глобальной не изменяется. Такой принцип называется областью видимости переменных (см. программный код ниже).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Глобальные и локальные переменные</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<script type="text/javascript">
function f_Sum() { var Var1 = 5;
var Num1 = 1;
document.write("Локальная переменная Var1 = " + Var1 + "<br>");
document.write("Локальная переменная Num1 = " + Num1 + "<br>");
document.write("Глобальная переменная Var2 = " + Var2 + "<br>");
return Var1+Var2; }
</script>
</head>
<body>
<script type="text/javascript">
var Var1, Var2, Var3;
Var1 = 10;
document.write("Глобальная переменная Var1 = " + Var1 + "<br>");
Var2 = 7;
Var3 = f_Sum();
document.write("Сумма Var1 + Var2 = " + Var3 + "<br>");
```

```
document.write("Глобальная переменная Var1 осталась = ");
document.write(Var1 + "<br>");
document.write("Локальная переменная Num1 = " + typeof Num1);
document.write(", т.е. не видна вне тела функции");
</script>
</body>
</html>
```

В окне веб-браузера будет получен следующий результат:

Глобальная переменная Var1 = 10
Локальная переменная Var1 = 5
Локальная переменная Num1 = 1
Глобальная переменная Var2 = 7
Сумма Var1 + Var2 = 12
Глобальная переменная Var1 осталась = 10

Локальная переменная Num1 = undefined, т.е. не видна вне тела функции. Глобальная переменная Var2 видна внутри функции f_Sum(). Глобальную переменную Var1 не затронуло объявление внутри функции локальной переменной Var1 и ее изменение.

7.15. Условные операторы

Условные операторы позволяют в зависимости от значения логического выражения выполнить отдельную ветвь программы или наоборот не выполнять ее. Логические выражения возвращают только два значения: true (истина) или false (ложь).

7.16. Операторы сравнения

В JavaScript в логических выражениях используются следующие операторы сравнения:

- == – равно;
- === – строго равно;
- != – не равно;
- !== – строго не равно;

- < – меньше;
- > – больше;
- <= – меньше или равно;
- >= – больше или равно.

Отличие оператора == (равно) от оператора === (строго равно) состоит в том, что если используется оператор ==, интерпретатор пытается преобразовать разные типы данных к одному и лишь затем сравнивает их. Оператор ===, встретив данные разных типов, сразу возвращает false (ложь).

Кроме того, значение логического выражения можно инвертировать с помощью оператора !, например, таким образом: *!(Var1 == Var2)*

Если переменные Var1 и Var2 равны, то возвращается значение true, но так как перед выражением стоит оператор !, выражение вернет false.

Несколько логических выражений можно объединить в одно большое с помощью следующих операторов:

- && – логическое И;
- || – логическое ИЛИ.

Например:

(Var1 == Var2) && (Var2 != Var3)
(Var1 == Var2) || (Var3 == Var4)

Первое выражение возвращает true только в случае, если оба выражения вернут true, а второе – если хотя бы одно из выражений вернет true.

Оператор || также часто используется для создания необязательных параметров в функции. Если первое выражение не может быть преобразовано в true, то возвращается значение второго выражения. Например:

```
function f_print(str) { str = str || "Значение по умолчанию";
window.alert(str); }
f_print(); // "Значение по умолчанию"
f_print("Значение указано"); // "Значение указано"
```

7.17. Оператор ветвления if…else. Проверка ввода пользователя

Оператор ветвления уже использовался ранее в рассмотренных ранее примерах. Например, чтобы проверить, какая из кнопок диалогового окна

нажата. Так как при нажатии кнопки OK возвращается значение true, то можно узнать, какая кнопка нажата, используя оператор ветвления if…else (см. программный код ниже).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Окно с сообщением</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript">
if (window.confirm("Нажмите любую кнопку")) {
window.alert("Нажата кнопка OK"); }
else { window.alert("Нажата кнопка Cancel"); }
</script>
</body>
</html>
```

В этом примере логическое выражение не содержит операторов сравнения: *if (window.confirm("Нажмите любую кнопку")) {*

Такая запись эквивалентна записи:

if (window.confirm("Нажмите любую кнопку") == true) {

Проверка на равенство выражения значению true (истина) выполняется по умолчанию.

Оператор ветвления if…else имеет следующий формат:

```
if (<Логическое выражение>) {
<Блок, выполняемый, если условие истинно> }
[else { <Блок, выполняемый, если условие ложно> }]
```

Ниже представлен пример программы, которая проверяет, является ли введенное пользователем число четным. После проверки выводится соответствующее сообщение.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Проверка числа на четность</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript">
var x = window.prompt("Введите число", "");
if (x==null) { document.write("Вы нажали Отмена"); }
else { if ((parseInt(x))%2==0) {
document.write("Четное число"); }
else { document.write("Нечетное число"); } }
</script>
</body>
</html>
```

Как видно из примера, один условный оператор можно вложить в другой.

Кроме того, если блок состоит из одного выражения, фигурные скобки можно не указывать:

```
if ((parseInt(x))%2==0) document.write("Четное число");
else document.write("Нечетное число");
```

Более того, блока else может не быть совсем:

```
if ((parseInt(x))%2==0) document.write("Четное число");
```

7.18. Оператор ?. Проверка числа на четность

Оператор ? имеет следующий формат:

```
<Переменная> = (<Лог. выражение>) ? <если Истина> : <если Ложь>;
```

Ниже представлен пример программы, проверяющей число на четность, с использованием оператора ?.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Проверка числа на четность</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript">
var x = window.prompt("Введите число", "");
if (x==null) { document.write("Вы нажали Отмена"); }
else { var msg = ((parseInt(x))%2==0) ? "Четное число" : "Нечетное число";
document.write(msg); }
</script>
</body>
</html>
```

7.19. Оператор выбора switch

Оператор выбора switch имеет следующий формат:

```
switch (<Переменная или выражение>) {
case <Значение 1>: <Выражение 1>; break;
case <Значение 2>: <Выражение 2>; break; ...
default: <Выражение>; }
```

Ниже представлен пример программы, проверяющей число на четность, с использованием оператора switch.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Проверка числа на четность</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
```

```
<body>
<script type="text/javascript">
var x = window.prompt("Введите число", "");
if (x==null) { document.write("Вы нажали Отмена"); }
else { switch ((parseInt(x))%2) { case 0: document.write("Четное число");
break;
case 1: document.write("Нечетное число");
break;
default: document.write("Введенное значение не является числом"); } }
</script>
</body>
</html>
```

Таким образом, оператор switch позволил сделать дополнительную проверку. Поскольку пользователь вместо числа мог ввести строку. А в этом случае функция parseInt() вернет значение NaN (Not a Number). Любая арифметическая операция со значением NaN вернет в качестве значения NaN. В предыдущих примерах такая проверка не выполнялась и в случае ввода строки, которую невозможно преобразовать в число, функция возвращала фразу "Нечетное число".

Вместо логического выражения оператор switch принимает переменную или выражение. В зависимости от значения переменной (или выражения) выполняется один из блоков case, в котором указано это значение. Если ни одно из значений не описано в блоках case, то выполняется блок default. Оператор break позволяет досрочно выйти из оператора выбора switch. Если не указать оператор break в конце блока case, то будет выполняться следующий блок case вне зависимости от указанного значения. Если убрать все операторы break из последнего примера, то в результате (при вводе четного числа) в окне веб-браузера отобразится следующее сообщение: *Четное число Нечетное число Введенное значение не является числом.* Т.е. оператор break следует обязательно указывать в конце каждого блока case.

7.20. Операторы циклов

Цикл for

Цикл for используется для выполнения выражений определенное число раз. Он имеет следующий формат:

for (<Начальное значение>; <Условие>; <Приращение>)
{ <Выражения> }

Здесь используются следующие конструкции:

• <Начальное значение> присваивает переменной-счетчику начальное значение;

• <Условие> содержит логическое выражение. Пока логическое выражение возвращает значение true, выполняются выражения внутри цикла;

• <Приращение> задает изменение переменной-счетчика при каждой итерации.

Более формально последовательность работы цикла for такова:

1. Переменной-счетчику присваивается начальное значение.
2. Проверяется условие и если оно истинно, выполняются выражения внутри цикла, а в противном случае осуществляется выход из цикла.
3. Переменная-счетчик изменяется на величину, указанную в <Приращении>.
4. Осуществляется переход к пункту 2.

Цикл выполняется до тех пор, пока <Условие> не вернет false. Если этого не случится, цикл будет бесконечным.

<Приращение> может не только увеличивать значение переменной-счетчика, но и уменьшать. Например, чтобы вывести все числа от 100 до 1, следует написать такую строку кода:

```
for (var i=100; i>0; i--) document.write(i + "<br>");
```

<Приращение> может изменять значение переменной-счетчика не только на единицу. Например, чтобы вывести все четные числа от 1 до 100, следует написать такую строку кода:

```
for (var i=2; i<101; i+=2) document.write(i + "<br>");
```

Выражение, указанное в параметре <Условие>, вычисляется на каждой итерации. Ниже представлен пример кода для вывода элементов массива:

```
var Mass = [1, 2, 3];
for (var i=0; i<Mass.length; i++) {
if (i==0) {
Mass.push(4); // Добавляем новые элементы
Mass.push(5); // для доказательства }
document.write(Mass[i] + " "); } // Выведет: 1 2 3 4 5
```

В этом примере в параметре <Условие> было указано свойство length, а внутри цикла (чтобы доказать вычисление на каждой итерации) в массив были добавлены новые элементы. В итоге были получены все элементы массива, включая новые элементы. Чтобы этого избежать следует вычисление размера массива указать в первом параметре:

```
var Mass = [1, 2, 3];
for (var i=0, c=Mass.length; i<c; i++) {
if (i==0) {
Mass.push(4); // Добавляем новые элементы
Mass.push(5); // для доказательства }
document.write(Mass[i] + " "); } // Выведет: 1 2 3
```

Цикл while

Выполнение выражений в цикле while продолжается до тех пор, пока логическое выражение истинно. Этот цикл имеет следующий формат:

```
<Начальное значение>;
while (<Условие>) {
<Выражения>;
<Приращение>; }
```

Цикл while работает следующим образом:

1. Переменной-счетчику присваивается начальное значение.
2. Проверяется условие, и если оно истинно, выполняются выражения внутри цикла, в противном случае выполнение цикла завершается.
3. Переменная-счетчик изменяется на величину, указанную в <Приращении>.
4. Осуществляется переход к пункту 2.

Ниже представлен пример кода, выводящего все числа от 1 до 100,

используя цикл while.

```
var i = 1;
while (i<101) { document.write(i + "<br>"); i++; }
```

Если <Приращение> не указано, то цикл будет бесконечным.

В <Приращении> необязательно должна быть арифметическая операция. Например, при работе с базами данных в качестве <Приращения> может бать использовано перемещение к следующей строке, а условием выхода из цикла – отсутствие новых строк в базе данных. В этом случае <Начальным значением> будет первая строка базы данных.

Цикл do…while

Выполнение выражений в цикле do…while продолжается до тех пор, пока логическое выражение истинно. Но в отличие от цикла while условие проверяется не в начале цикла, а в конце. Поэтому выражения внутри цикла do…while один раз обязательно выполнятся. Конструкция имеет следующий формат:

```
<Начальное значение>;
do {
<Выражения>;
<Приращение>; }
while (<Условие>);
```

Последовательность работы цикла do…while:

1. Переменной-счетчику присваивается начальное значение.
2. Выполняются выражения внутри цикла.
3. Переменная-счетчик изменяется на величину, указанную в <Приращении>.
4. Проверяется условие, и если оно истинно, осуществляется переход к пункту 2, а если нет – цикл завершается.

Ниже представлен пример кода, выводящего все числа от 1 до 100, используя цикл do…while.

```
var i = 1;
do { document.write(i + "<br>"); i++; } while (i<101);
```

Если <Приращение> не указано, то цикл будет бесконечным.

Оператор continue. Переход на следующую итерацию цикла

Оператор continue позволяет перейти на следующую итерацию цикла еще до завершения выполнения всех выражений внутри цикла. Этот оператор можно применять в любых циклах.

Ниже представлен пример кода, выводящего все числа от 1 до 100, кроме чисел от 5 до 10 включительно.

```
for (var i=1; i<101; i++) { if (i>4 && i<11) continue;
document.write(i + "<br>"); }
```

Оператор break. Прерывание цикла

Оператор break позволяет прервать выполнение цикла досрочно. Ниже представлен пример кода, выводящего все числа от 1 до 100, с использованием оператора break.

```
for (var i=1; true; i++) {
if (i>100) break;
document.write(i + "<br>"); }
```

В этом примере было указано условие продолжения цикла, которое всегда истинно, и без использования оператора break цикл продолжался бы бесконечно.

Оператор break прерывает выполнение цикла, а не программы, то есть далее будет выполнено выражение, следующее за циклом.

7.21. Ошибки в программе

Существуют три типа ошибок в скриптах: синтаксические, логические и ошибки времени выполнения.

Синтаксические ошибки – это ошибки в синтаксисе языка, например, ошибки в имени оператора или функции, отсутствие или наличие лишней закрывающей или открывающей скобок, буква набрана в русской раскладке клавиатуры вместо латинской, неправильный регистр букв, в цикле for указаны параметры через запятую, а не через точку с запятой и т.д. Как правило, интерпретатор предупреждает о наличии ошибки, а программа не будет выполняться.

Например, если вместо *document.write(i + "
");* написать

*doument.write(i + "
");* то веб-браузер отобразит такое сообщение:

Error:

name: ReferenceError

message: Statement on line 5: Reference to undefined variable: doument

Backtrace:

Line 5 of inline#1 script in test.html

*doument.write(i + "
");*

Т.е. веб-браузер предупредит, что в строке 5 файла test.html есть ошибка.

Логические ошибки – это ошибки в логике работы программы, которые можно выявить только по результатам работы скрипта. Как правило, интерпретатор не предупреждает о наличии ошибки, и программа будет выполняться, так как не содержит синтаксических ошибок. Такие ошибки трудно выявить и исправить.

Например, если для вывода первых трех элементов массива использовать следующий код:

```
var Mass1 = [1, 2, 3, 4];
for (var i=1; i<4; i++) document.write(Mass1[i]+ "<br>");
```

Возникнет логическая ошибка, так как будут получены не первые элементы массива, а три элемента, начиная со второго. А поскольку индексация массивов начинается с нуля, в данном примере нет синтаксических ошибок, интерпретатор сочтет код правильным.

Логическая ошибка возникнет и в том случае, если, например, в логическом выражении вместо оператора == (равно) указать оператор присваивания =:

```
var X = 5;
if (X=6) document.write("Переменная X равна 6");
else document.write("Переменная X НЕ равна 6");
```

В результате будет выведено сообщение:

Переменная X равна 6

Ошибки времени выполнения – это ошибки, которые возникают во время работы скрипта. Причиной их являются не предусмотренные разработчиком события.

В некоторых языках ошибки времени выполнения возникают, например, из-за деления на ноль или обращения к несуществующему элементу массива. В языке JavaScript в этих случаях программа прервана не будет. И при попытке деления на ноль возвращается значение Infinity. Например:
window.alert(5/0); // Infinity

А при обращении к несуществующему элементу массива возвращается значение undefined. Например:
var arr = [1, 2];
window.alert(arr[20]); // undefined

Очень часто ошибки времени выполнения возникают при использовании условий. Например:
if (x>5) window.alert("x > 5");
*else doument.write(x + "
"); // Строка с ошибкой*

В этом примере ошибки не будет, пока соблюдается условие "x>5". Как только условие перестанет выполняться, возникнет ошибка, и выполнение программы будет прервано.

7.22. Обработка ошибок

Перехватить и обработать ошибки позволяет конструкция try/catch/finally. Конструкция имеет следующий формат:
try {<Выражения, в которых перехватываем ошибки> }
[catch ([<Ссылка на объект Error>]) { <Обработка ошибки> }]
[finally { <Выражения, которые будут выполнены в любом случае> }]

Выражения, в которых могут возникнуть ошибки, размещаются в блоке try. Если внутри этого блока возникнет исключение, то управление будет передано в блок catch. В качестве параметра в блоке catch можно указать переменную, через которую будет доступен объект Error, содержащий описание ошибки. Если в блоке try ошибки не возникло, то блок catch не выполняется. Если указан блок finally, то выражения внутри этого блока будут выполнены независимо от того, возникла ошибка или нет. Блоки catch и finally являются необязательными, но хотя бы один из них должен быть указан.

В некоторых случаях требуется не обрабатывать ошибку, а, наоборот, указать программе, что возникла неисправимая ошибка, и прервать выполнение всей программы. Для этого предназначен оператор throw. Например:

```
if (d < 0)
throw new Error("Переменная не может быть меньше нуля");
```

7.23. Модуль Firebug для веб-браузера Firefox

Firebug – это модуль для веб-браузера Firefox, предназначенный для отладки веб-страниц и скриптов. Загрузить модуль можно с сайта разработчика http://getfirebug.com/ или со страницы https://addons.mozilla.org/ru/firefox/addon/1843.

На вкладке HTML отображается весь код страницы. При наведении курсора мыши на определенный тег элемент подсвечивается на веб-странице, а справа на вкладке Макет видна структура блочной модели со значениями атрибутов margin, border и padding (рис. 7.1). Значения этих атрибутов можно изменять и одновременно наблюдать за результатом произведенных изменений.

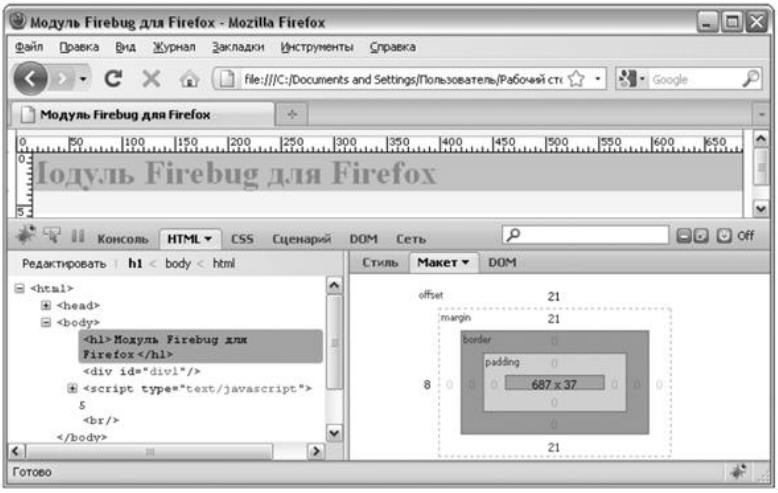


Рис. 7.1. Структура блочной модели, отображаемая на вкладке Макет

На вкладке Сеть отображается весь процесс загрузки веб-страницы. Можно узнать скорость загрузки отдельных компонентов, а также посмотреть HTTP-заголовки запроса веб-браузера и HTTP-заголовки ответа сервера.

Модуль Firebug также можно использовать для поиска ошибок в скриптах. Если ввести строку с ошибкой, например: *doument.write(i + "
");* , то после загрузки страницы на вкладке Консоль появится сообщение об ошибке (рис. 7.2). При этом текст ошибки будет ссылкой, при переходе по которой станет активной вкладка Сценарий, а строка с ошибкой некоторое время будет подсвечена.

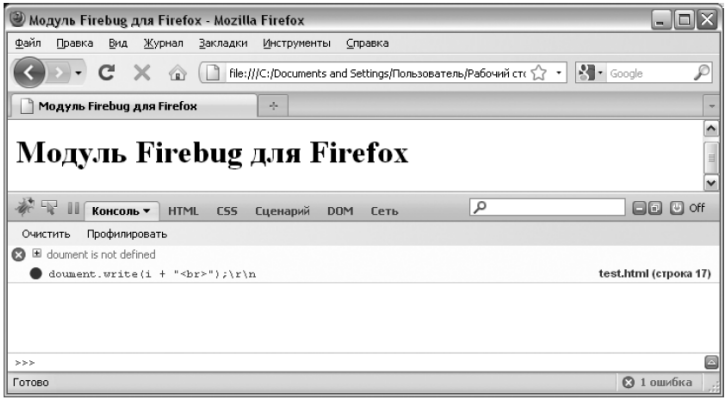


Рис. 7.2. Сообщение об ошибке, выводимое модулем Firebug

Вкладка Сценарий является полноценным отладчиком скриптов на JavaScript. Здесь можно установить точки остановки. Для этого необходимо щелкнуть левой кнопкой мыши справа от номера строки. В результате должна появиться красная точка. Теперь после обновления веб-страницы программа прервется на отмеченной строке (рис. 7.3). В этот момент можно посмотреть текущие значения переменных, а также продолжить выполнение скрипта по шагам. Таким образом, можно полностью контролировать весь процесс выполнения программы.

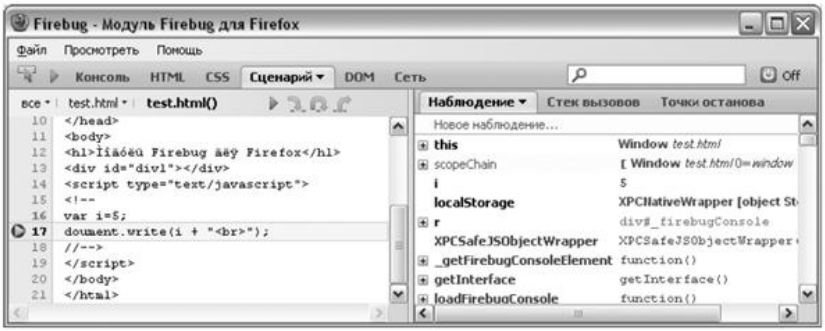


Рис. 7.3. Пошаговое выполнение программы в Firebug

В веб-браузере Internet Explorer 8.0 имеется аналогичный модулю Firebug инструмент. Он называется "Средства разработчика". Для его запуска в меню Сервис следует выбрать пункт Средства разработчика или нажать клавишу F12. В открывшемся окне можно просматривать структуру HTML и CSS (рис. 7.4) и отлаживать скрипты (рис. 7.5).

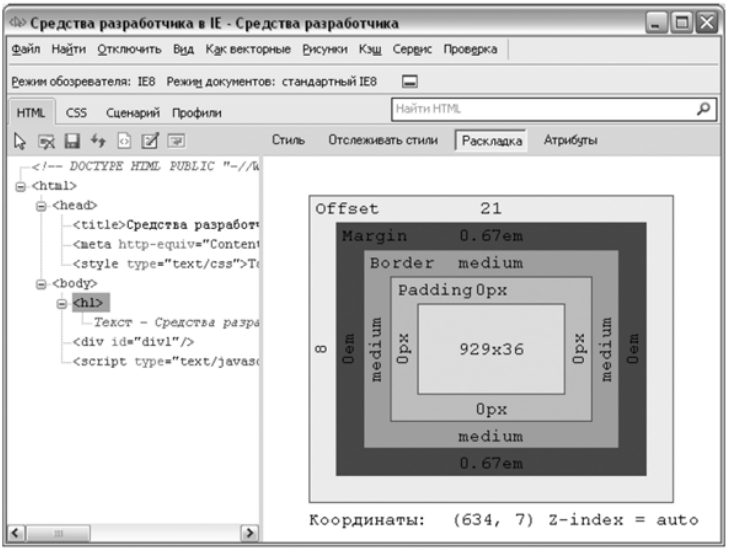


Рис. 7.4. Окно Средства разработчика в веб-браузере Internet Explorer 8.0

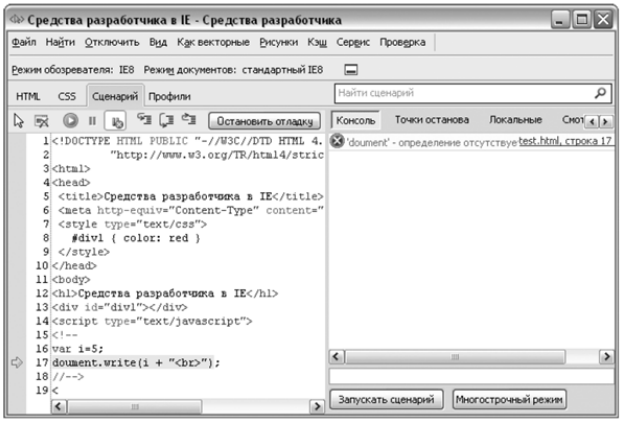


Рис. 7.5. Отладка скриптов в веб-браузере Internet Explorer 8.0

Практические задания

Вариант 1

1. Используя язык сценария JavaScript на созданной HTML – странице отобразить кнопку, при нажатии на которую на экран выводится окно, с предложением ввести сегодняшнюю температуру в градусах по Фаренгейту.

После чего на экран выводится окно с указанием величины соответствующей температуры в градусах по Цельсию.

Для перевода используйте следующую формулу:

Температура по Фаренгейту=1.8*температура по Цельсию + 32

Если температура по Цельсию меньше нуля, выводится сообщение "Не забудьте надеть шапку! Берегите голову!". Если температура по Цельсию больше нуля, но меньше 28, выводится сообщение "Идите на улицу спокойно, ни о чем не беспокойтесь!". И если температура по Цельсию больше 28 градусов, выводится сообщение "Не забудьте надеть панаму, а то будет солнечный удар!".

2. Написать скрипт, который будет выполняться по щелчку на кнопке. Реализуйте с помощью функции на языке JavaScript следующую задачу: дан целочисленный массив размера N. Вывести вначале все содержащиеся

в данном массива четные числа в порядке возрастания их индексов, а затем – все нечетные числа в порядке убывания их индексов.

Вариант 2

1. С помощью метода prompt создайте пользовательскую функцию, которая будет реализовывать следующий диалог:

Программа: Здравствуйте! Как Ваше имя?

Пользователь: Иван

Программа: Доброе утро, Иван! Как настроение?

Пользователь: так себе

Программа: У меня тоже так себе, Иван!

2. Реализуйте с помощью функции на языке JavaScript следующую задачу: дан целочисленный массив размера N. Найти индекс и значение первого максимального нечетного числа из данного массива. Если нечетные числа отсутствуют, то вывести сообщение "Нечетные числа отсутствуют".

Вариант 3

1. Добавить на страницу гиперссылку, щелчком на которую выводится окно сообщения «Предполагаем радиус равен» и псевдослучайное число от нуля до ста. После закрытия окна сообщения выводится следующее окно, в котором отображается «Объем сферы с заданным радиусом равен » и высчитанный по соответствующей формуле объем сферы:

$$V = \frac{4}{3}\pi R^3.$$

2. Добавьте в документ рисунок и реализуйте так, чтобы скрипт срабатывал при наведении курсора на это изображение. А именно: выводилось окно с вопросом «Нравится ли Вам рисунок?» (предполагаемые варианты ответов должны отображаться в поле ввода). Если пользователь ответит «Да», то выводится сообщение «Мы рады, что Вы неравнодушны к искусству!». Если пользователь ответит «Нет», то выводится сообщение «Сожалеем, что спросили об этом!».

Вариант 4

1. Создайте пользовательскую функцию, которая будет запрашивать логин пользователя (метод prompt()) и проверять существует ли данный логин среди уже зарегистрированных (реализовать с помощью массива). В результате работы программы, если такой логин уже есть, то выводится сообщение «Приветствуем Вас на нашем сайте, логин_пользователя!». Если такого логина нет, то вывести сообщение «Вы не зарегистрированы на нашем сайте!».

2. Реализуйте с помощью функции на языке JavaScript следующую задачу: робот может перемещаться в четырех направлениях («С» – север, «З» – запад, «Ю» – юг, «В» – восток) и выполнять четыре команды: вперед – продолжить движение, налево – повернуть налево, направо – повернуть направо; назад – двигаться в противоположном направлении. Пользователь вводит исходное направление робота и команду. Вывести направление движения робота после выполнения полученной команды.

Вариант 5

1. Написать скрипт, реализующий сложение, вычитание, умножение и деление двух чисел по щелчку на кнопке. Оба числа вводятся пользователем через диалоговые окна, а затем выводятся на экран в формате, например, 5+4=9, 5-4=1, 5*4=20, 5/4=1.25

2. Реализуйте с помощью функции на языке JavaScript следующую задачу: пользователь вводит два числа: D (день) и M (месяц), определяющие дату рождения. Вывести знак зодиака и характеристику, соответствующую этому знаку зодиака: Овен (21.03 – 20.04), Телец (21.04 – 20.05), Близнецы (21.05 – 21.06), Рак (22.06 – 22.07), Лев (23.07 – 23.08), Дева (24.08 – 23.09), Весы (24.09 – 23.10), Скорпион (24.10 – 22.11), Стрелец (23.11 – 21.12), Козерог (22.12 – 20.01), Водолей (21.01 – 20.02), Рыбы (21.02 – 20.03).

Контрольные вопросы

1. Каким образом можно разместить код JavaScript на HTML-странице?
2. Как добавить комментарий в код JavaScript?
3. Для чего используется оператор var?
4. Какие операторы существуют в JavaScript?
5. Что такое блок try/catch/finally?
6. Способы создания переменной?
7. Какие типы переменных существуют в JavaScript?
8. Какие арифметические операции существуют в JavaScript?
9. Синтаксис полного оператора if.
10. Синтаксис сокращённого оператора if.
11. В каком случае можно не использовать операторные скобки в операторе if?
12. Синтаксис оператора цикла for.
13. Будет ли ошибкой, если в теле оператора цикла не будет ни одного оператора?
14. Чему равна переменная-параметр в цикле for после завершения оператора цикла?
15. Приведите пример создания функции.
16. Перечислите формальные параметры функции.
17. Какова локальная область видимости переменных?
18. Как возвратить результат из функции?
19. Как передать из функции несколько значений переменных?

8. РАБОТА СО ВСТРОЕННЫМИ КЛАССАМИ JAVASCRIPT. СОЗДАНИЕ ДИНАМИЧЕСКИХ ПРИЛОЖЕНИЙ

8.1. Встроенные классы JavaScript

Класс – это тип объекта, включающий в себя переменные и управляющие ими функции для управления этими переменными. Переменные называют свойствами, а функции – методами. Для использования методов и свойств класса чаще всего необходимо создать экземпляр класса. Для этого используется оператор new, после него указывается имя класса, к которому будет относиться данный экземпляр. После имени класса, в круглых скобках, можно задавать параметры, устанавливая таким образом начальные значения для свойств класса:

<Экземпляр класса> = new <Имя класса> ([<Параметры>]);

При создании экземпляра класса ссылка (указатель) сохраняется в переменной. Используя ссылку, можно обращаться к свойствам и методам созданного экземпляра класса.

При обращении к свойствам используется следующий формат:

<Экземпляр класса>.<Имя свойства>;

Обращение к методам осуществляется аналогично, только после имени метода необходимо поставить круглые скобки:

<Экземпляр класса>.<Имя метода>();

В скобках часто указываются параметры метода.

8.2. Класс Global

Свойства и методы класса Global являются встроенными функциями JavaScript. Использование свойств и методов данного класса не требует создания экземпляра класса. Свойства класса Global:

- NaN содержит значение Not a Number. Например: *var x = NaN*;
- Infinity возвращает значение "плюс бесконечность". Например: *var x = Infinity*;

Методы класса Global:

- parseInt(<Строка>, [<Основание>]) преобразует строку в целое чис-

ло системы счисления, заданной основанием. Если основание не указано, то по умолчанию используется десятичная система. Если строка не может быть преобразована в число, возвращается значение NaN. Например:

```
var Number1 = 15;
var Str = "5";
var Str5 = "FF";
var Str2 = Number1 – parseInt(Str); // Переменная содержит число 10
var Str3 = Number1 – parseInt(Str5, 16); // Переменная содержит число –240
var Str4 = Number1 + parseInt(Str); // Переменная содержит число 20
```

• parseFloat(<Строка>) преобразует строку в число с плавающей точкой. Например:

```
var Str = "5.2";
var Str2 = parseFloat(Str); // Переменная содержит число 5.2
```

• eval(<Строка>) вычисляет выражение в строке, как если бы это было обычное выражение JavaScript:

```
var Str = "3 + 5"; var Str2 = eval(Str); // Переменная содержит число 8
```

• isNaN(<Выражение>) проверяет, является ли выражение правильным числом. Возвращает true, если значение выражения равно NaN, и false, если выражение возвращает число;

• isFinite(<Выражение>) проверяет, является ли выражение конечным числом. Возвращает true или false;

• escape(<Строка>) кодирует строку шестнадцатеричными кодами. Например:

```
var Str = escape("Привет");
// Str = %u041F%u0440%u0438%u0432%u0435%u0442
```

• unescape(<Строка>) декодирует строку, закодированную методом escape(). Например:

```
var Str = unescape
("%u041F%u0440%u0438%u0432%u0435%u0442"); // Str = Привет
```

• encodeURI(<URL-адрес>) кодирует URL-адрес целиком. Например:

```
var Str = "test.php?id=5&n=Николай";
window.alert(encodeURI(Str));
```

//test.php?id=5&n=%D0%9D%D0%B8%D0%BA%D0%BE%D0%BB%D0%B0%D0%B9

• decodeURI(<Строка>) декодирует строку, закодированную методом encodeURI();

• encodeURIComponent(<Строка>) выполняет URL-кодирование строки. Например:

var Str = encodeURIComponent("Строка");
// Str = %D0%A1%D1%82%D1%80%D0%BE%D0%BA%D0%B0

В отличие от функции encodeURI() функция encodeURIComponent() заменяет все спецсимволы шестнадцатеричными кодами:

var Str = "test.php?name=Николай";
window.alert(encodeURIComponent(Str));
//test.php%3Fname%3D%D0%9D%D0%B8%D0%BA%D0%BE%D0%BB%D0%B0%D0%B9

• decodeURIComponent(<Строка>) декодирует строку, закодированную методом encodeURIComponent().

8.3. Класс Number. Работа с числами

Класс Number используется для хранения числовых величин, а также для доступа к константам. Экземпляр класса создается по такой схеме:

<Экземпляр класса> = new Number (<Начальное значение>);

Свойства класса Number можно использовать без создания экземпляра класса:

▪ MAX_VALUE – максимально допустимое в JavaScript число. Например: *var x = Number.MAX_VALUE; // 1.7976931348623157e+308*

▪ MIN_VALUE – минимально допустимое в JavaScript число. Например: *var x = Number.MIN_VALUE; // 5e-324*

▪ NaN – значение NaN. Например: *var x = Number.NaN; // NaN*

▪ NEGATIVE_INFINITY – значение "минус бесконечность". Например: *var x = Number.NEGATIVE_INFINITY; // -Infinity*

▪ POSITIVE_INFINITY – значение "плюс бесконечность". Например: *var x = Number.POSITIVE_INFINITY; // Infinity*

Методы класса Number:

- valueOf() возвращает числовое значение экземпляра класса. Например:

```
var x = new Number (15);
var y = x.valueOf(); // 15
document.write(typeof y); // number
```

- toString() возвращает строковое представление числа. Например:

```
var x = new Number (15);
var Str = x.toString(); // "15"
document.write(typeof Str); // string
```

8.4. Класс String. Обработка строк

Класс String предоставляет доступ к множеству методов для обработки строк. Экземпляр класса создается по следующей схеме:

<Экземпляр класса> = new String (<Строка>);

Создать строку можно с помощью двойных или одинарных кавычек. Например:

```
var Str1 = "Строка 1";
var Str2 = 'Строка 2';
```

Строки, созданные этими способами, будут иметь тип данных string, а при создании экземпляра класса String тип данных будет object. Например:

```
var Str1 = "Строка 1";
var Str2 = 'Строка 2';
var Str3 = new String ("Строка 3");
document.write(typeof Str1); // string
document.write(typeof Str2); // string
document.write(typeof Str3); // object !
```

Тем не менее, к обычным строкам можно также применять методы класса String. Например:

```
var Str = "Строка".toUpperCase(); // Перевод символов в верхний регистр
document.write(Str); // "СТРОКА"
document.write(typeof Str); // string
```

При использовании метода toUpperCase() строка, имеющая тип дан-

ных string, автоматически преобразуется в экземляр класса String. Затем производится изменение (в примере, представленном выше, перевод символов в верхний регистр) и возвращается строка, имеющая тип данных string. Таким образом, класс String является объектом-оберткой над элементарным типом данных string.

Свойство length возвращает длину строки в символах. Например:

```
var Str = new String ("Hello, world");
document.write(Str.length); // 12
```

Методы класса String:

• toString() и valueOf() возвращают значение строки. Например:

```
var Str = new String ("Hello, world");
var Str2 = Str.toString();
document.write(Str2); // "Hello, world"
document.write(typeof Str); // object
document.write(typeof Str2); // string
```

• charAt(<Номер символа>) извлекает символ, номер которого указан в качестве параметра. Нумерация символов в строке начинается с нуля. Например:

```
var Str = "Hello, world";
document.write(Str.charAt(0)); // "H"
```

• charCodeAt(<Номер символа>) возвращает код символа, номер которого указан в качестве параметра. Нумерация символов в строке начинается с нуля. Например:

```
var Str = "Hello, world";
window.alert(Str.charCodeAt(0)); // 72
```

• fromCharCode(<Код1>,...,<КодN>) создает строку из указанных кодов. Например:

```
var S = String.fromCharCode(1055, 1088, 1080, 1074, 1077, 1090);
window.alert(S); // "Привет"
```

• toLowerCase() преобразует символы строки в символы нижнего регистра. Например:

```
var Str = "Hello, world";
```

Str = Str.toLowerCase();
document.write(Str); // "hello, world"

• toUpperCase() преобразует символы строки в символы верхнего регистра. Например:

var Str = "Hello, world";
Str = Str.toUpperCase();
document.write(Str); // "HELLO, WORLD"

• substr(<Начало фрагмента>, [<Длина фрагмента>]) извлекает фрагмент строки заданной длины. Если второй параметр пропущен, возвращаются все символы до конца строки. Например:

var Str = "Hello, world";
document.write(Str.substr(0, 5)); // "Hello"
document.write(Str.substr(7)); // "world"

• substring(<Начало фрагмента>, <Конец фрагмента>) также извлекает фрагмент строки, заданный в этом случае номерами начального и конечного символов. Последний символ во фрагмент не включается. Например:

var Str = "Hello, world";
document.write(Str.substring(7, 12)); // "world"

• indexOf(<Подстрока>, [<Начальная позиция поиска>]) возвращает номер позиции первого вхождения подстроки в текущей строке. Если второй параметр не задан, то поиск начинается с начала строки. Если подстрока не найдена, возвращается значение −1. Например:

var Str = "Hello, world";
document.write(Str.indexOf("ll")); // 2
document.write(Str.indexOf("ll", 5)); // –1

• lastIndexOf(<Подстрока>, [<Начальная позиция поиска>]) определяет номер позиции последнего вхождения подстроки в текущей строке. Если второй параметр не задан, то поиск начинается с начала строки. Если подстрока не найдена, возвращается значение −1. Например:

var Str = "Hello, world";
document.write(Str.lastIndexOf("o")); // 8

• split(<Разделитель>, [<Лимит>]) возвращает массив, полученный в

результате разделения строки на подстроки по символу-разделителю. Если второй параметр присутствует, то он задает максимальное количество элементов в результирующем массиве. Например:

```
var Str = "Hello, world";
var Mass = Str.split(",");
document.write(Mass[0]); // "Hello" – первый элемент массива
document.write(Mass[1]); // " world" – второй элемент массива
```

• search(<Регулярное выражение>) определяет номер позиции первого вхождения подстроки, совпадающей с регулярным выражением;

• match(<Регулярное выражение>) возвращает массив с результатами поиска, совпадающими с регулярным выражением;

• replace(<Регулярное выражение>, <Текст для замены>) возвращает строку, которая является результатом поиска и замены в исходной строке с использованием регулярного выражения.

Примеры использования последних трех методов будут рассмотрены при изучении регулярных выражений и встроенного класса RegExp.

8.5. Класс Array. Работа с массивами и их сортировка

Класс Array позволяет создавать массивы как объекты и предоставляет доступ к множеству методов для обработки массивов. Экземпляр класса можно создать следующими способами:

1. <Экземпляр класса> = new Array (<Количество элементов массива>).

2. <Экземпляр класса> = new Array (<Элементы массива через запятую>).

Если в круглых скобках нет никаких параметров, то создается массив нулевой длины, то есть массив, не содержащий элементов. Если указано одно число, то это число задает количество элементов массива. Если указано несколько элементов через запятую или единственное значение не является числом, то указанные значения записываются в создаваемый массив.

Обращение к элементам массива осуществляется с помощью квадратных скобок, в которых указывается индекс элемента. Нумерация элементов массива начинается с нуля. Например:

```
var Mass = new Array("Один", "Два", "Три");
```

```
document.write(Mass[0]); // "Один"
Mass[3] = 4; // Создание нового элемента массива
document.write(Mass.join(", ")); // "Один, Два, Три, 4"
```

Свойство length возвращает количество элементов массива. Например:

```
var Mass = [ "Один", "Два", "Три" ];
document.write(Mass.length + "<br>"); // 3
// Выводим все элементы массива по одному на строку
for (var i=0, c=Mass.length; i<c; i++) {
document.write(Mass[i] + "<br>"); }
```

Работу с массивами обеспечивают следующие методы:

- push(<Список элементов>) добавляет в массив элементы, указанные в списке элементов. Элементы добавляются в конец массива. Метод возвращает новую длину массива. Например:

```
var Mass = [ "Один", "Два", "Три" ];
document.write(Mass.push("Четвертый", "Пятый")); // 5
document.write(Mass.join(", ")); // "Один, Два, Три, Четвертый, Пятый"
```

- unshift(<Список элементов>) добавляет в массив элементы, указанные в списке элементов. Элементы добавляются в начало массива. Например:

```
var Mass = [ "Один", "Два", "Три" ];
Mass.unshift("Четвертый", "Пятый");
document.write(Mass.join(", ")); // "Четвертый, Пятый, Один, Два, Три"
```

- concat(<Список элементов>) возвращает массив, полученный в результате объединения текущего массива и списка элементов. При этом в текущий массив элементы из списка не добавляются. Например:

```
var Mass = [ "Один", "Два", "Три" ];
var Mass2 = []; // Пустой массив
Mass2 = Mass.concat("Четвертый", "Пятый");
document.write(Mass.join(", ")); // "Один, Два, Три"
document.write(Mass2.join(", ")); // "Один, Два, Три, Четвертый, Пятый"
```

- join(<Разделитель>) возвращает строку, полученную в результате объединения всех элементов массива через разделитель. Например:

```
var Mass = [ "Один", "Два", "Три" ];
```

var Str = Mass.join(" - ");
document.write(Str); // "Один – Два – Три"

▪ shift() удаляет первый элемент массива и возвращает его. Например:

var Mass = ["Один", "Два", "Три"];
document.write(Mass.shift()); // "Один"
document.write(Mass.join(", ")); // "Два, Три"

▪ pop() удаляет последний элемент массива и возвращает его. Например:

var Mass = ["Один", "Два", "Три"];
document.write(Mass.pop()); // "Три"
document.write(Mass.join(", ")); // "Один, Два"

▪ sort([Функция сортировки]) выполняет сортировку массива. Если функция не указана, будет выполнена обычная сортировка (числа сортируются по возрастанию, а символы – по алфавиту). Например:

var Mass = ["Один", "Два", "Три"];
Mass.sort();
document.write(Mass.join(", ")); // "Два, Один, Три"

Чтобы изменить стандартный порядок сортировки можно воспользоваться функцией сортировки, которая принимает две переменные и должна возвращать:

- 1 – если первый элемент массива больше второго;
- –1 – если второй элемент массива больше первого;
- 0 – если элементы массива равны.

Например, стандартная сортировка зависит от регистра символов:

var Mass = ["единица1", "Единый", "Единица2"];
Mass.sort();
document.write(Mass.join(", ")); // "Единица2, Единый, единица1"

В результате будет получена неправильная сортировка, поскольку "Единица2" и "Единый" должны стоять после элемента "единица1". Следовательно, необходимо изменить стандартную сортировку на собственную сортировку без учета регистра:

function f_sort(Str1, Str2) { // Сортировка без учета регистра
var Str1_1 = Str1.toLowerCase(); // Преобразуем к нижнему регистру

```
var Str2_1 = Str2.toLowerCase(); // Преобразуем к нижнему регистру
if (Str1_1>Str2_1) return 1;
if (Str1_1<Str2_1) return –1;
return 0; }
var Mass = [ "единица1", "Единый", "Единица2" ];
Mass.sort(f_sort); // Имя функции указывается без скобок
document.write(Mass.join(", ")); // "единица1, Единица2, Единый"
```

В этом примере две переменные сначала приводятся к одному регистру, а затем производится стандартное сравнение. При этом не изменяется регистр самих элементов массива, так как работа осуществляется с их копиями.

Порядок сортировки можно изменить на противоположный, изменив возвращаемые функцией значения. Например:

```
function f_sort(Str1, Str2) {
// Сортировка без учета регистра в обратном порядке
var Str1_1 = Str1.toLowerCase(); // Преобразуем к нижнему регистру
var Str2_1 = Str2.toLowerCase(); // Преобразуем к нижнему регистру
if (Str1_1>Str2_1) return –1;
if (Str1_1<Str2_1) return 1;
return 0; }
var Mass = [ "единица1", "Единица2", "Единый" ];
Mass.sort(f_sort);
document.write(Mass.join(", ")); // "Единый, Единица2, единица1"
```

- reverse() переворачивает массив. Элементы будут следовать в обратном порядке относительно исходного массива. Например:

```
var Mass = [ "Один", "Два", "Три" ];
Mass.reverse();
document.write(Mass.join(", ")); // "Три, Два, Один"
```

- slice(<Начало>, [<Конец>]) возвращает срез массива, начиная от индекса <Начало> и заканчивая индексом <Конец>, но не включает элемент с этим индексом. Если второй параметр не указан, то возвращаются все элементы до конца массива. Например:

```
var Mass1 = [ 1, 2, 3, 4, 5 ];
```

```
var Mass2 = Mass1.slice(1, 4);
window.alert(Mass2.join(", ")); // "2, 3, 4"
var Mass3 = Mass1.slice(2);
window.alert(Mass3.join(", ")); // "3, 4, 5"
```

▪ splice(<Начало>, <Количество>, [<Список значений>]) позволяет удалить, заменить или вставить элементы массива. Возвращает массив, состоящий из удаленных элементов. Например:

```
var Mass1 = [ 1, 2, 3, 4, 5 ];
var Mass2 = Mass1.splice(2, 2);
window.alert(Mass1.join(", ")); // "1, 2, 5"
window.alert(Mass2.join(", ")); // "3, 4"
var Mass3 = Mass1.splice(1, 1, 7, 8, 9);
window.alert(Mass1.join(", ")); // "1, 7, 8, 9, 5"
window.alert(Mass3.join(", ")); // "2"
var Mass4 = Mass1.splice(1, 0, 2, 3, 4);
window.alert(Mass1.join(", ")); // "1, 2, 3, 4, 7, 8, 9, 5"
window.alert(Mass4.join(", ")); // Пустой массив
```

▪ toString() и valueOf() преобразуют массив в строку. Элементы указываются через запятую без пробела. Например:

```
var Mass = [ "Один", "Два", "Три" ];
document.write(Mass.toString()); // "Один,Два,Три"
```

8.6. Многомерные массивы

Многомерные массивы можно задать:

1) перечислением. Например:

```
var Mass = new Array(new Array("Один", "Два", "Три"),
new Array("Четыре", "Пять", "Шесть"));
document.write(Mass[0][1]); // "Два"
var Mass2 = [ [ "Один", "Два", "Три" ],
[ "Четыре", "Пять", "Шесть" ] ];
document.write(Mass2[1][1]); // "Пять"
```

2) поэлементно. Например:

```
var Mass = new Array();
Mass[0] = new Array();
Mass[1] = new Array();
Mass[0][0] = "Один";
Mass[0][1] = "Два";
Mass[0][2] = "Три";
Mass[1][0] = "Четыре";
Mass[1][1] = "Пять";
Mass[1][2] = "Шесть";
document.write(Mass[1][2]); // "Шесть"
var Mass2 = [];
Mass2[0] = [];
Mass2[1] = [];
Mass2[0][0] = "Один";
Mass2[0][1] = "Два";
Mass2[0][2] = "Три";
Mass2[1][0] = "Четыре";
Mass2[1][1] = "Пять";
Mass2[1][2] = "Шесть";
document.write(Mass2[0][0]); // "Один"
```

Обращение к элементу многомерного массива осуществляется с помощью двух индексов. Например: *var Str = Mass[1][2];*

8.7. Ассоциативные массивы. Перебор ассоциативных массивов

Основным отличием ассоциативных массивов от обычных является возможность обращения к элементу массива не по числовому индексу, а по индексу, представляющему собой строку. Например:

```
var Mass = new Array();
Mass["Один"] = 1;
Mass["Два"] = 2;
Mass["Три"] = 3;
```

document.write(Mass["Один"]); // 1

Ни один из методов класса Array не позволяет вывести элементы ассоциативного массива. Свойство length также не работает. По этой причине перебрать все элементы массива с помощью стандартного цикла for не представляется возможным.

Для этого существует специальный цикл for…in. Он имеет следующий формат:

for (<Переменная> in <Экземпляр класса>) { <Тело цикла> }

Цикл for…in на каждой итерации присваивает <Переменной> имя свойства, с помощью которого можно получить значение соответствующего элемента ассоциативного массива. Например:

```
var Mass = new Array();
Mass["Один"] = 1;
Mass["Два"] = 2;
Mass["Три"] = 3;
for (var Name in Mass) {
// Переменной Name на каждой итерации присваивается строка-индекс ассоциативного массива
document.write(Name + " = " + Mass[Name] + "<br>"); }
```

В итоге будет получен следующий результат:

```
Один = 1
Два = 2
Три = 3
```

Ассоциативные массивы используются также для доступа к свойствам класса вместо классической точки. Для получения длины строки ранее мы обращались к свойству length класса String следующим образом:

```
var Str = "Hello, world ";
document.write(Str.length); // 13
```

С помощью ассоциативных массивов обращение к свойству length будет выглядеть так:

```
var Str = "Hello, world ";
document.write(Str["length"]); // 13
```

8.8. Класс Math. Использование математических функций

Класс Math содержит математические константы и функции. Его использование не требует создания экземпляра класса.

Свойства класса Math:

- E – e, основание натурального логарифма;
- LN2 – натуральный логарифм 2;
- LN10 – натуральный логарифм 10;
- LOG2E – логарифм по основанию 2 от e;
- LOG10E – десятичный логарифм от e;
- PI – число Пи (например, document.write(Math.PI); // 3.141592653589793)
- SQRT1_2 – квадратный корень из 0,5;
- SQRT2 – квадратный корень из 2.

Методы класса Math:

- abs() – абсолютное значение;
- sin(), cos(), tan() – стандартные тригонометрические функции (синус, косинус, тангенс). Значение указывается в радианах;
- asin(), acos(), atan() – обратные тригонометрические функции (арксинус, арккосинус, арктангенс). Значение возвращается в радианах;
- exp() – экспонента;
- log() – натуральный логарифм;
- pow(<Число>, <Степень>) – возведение <Числа> в <Степень>. Например:

var x = 5; document.write(Math.pow(x, 2)); // 25 (5 в квадрате)

- sqrt() – квадратный корень: Например:

var x = 25; document.write(Math.sqrt(x)); // 5 (квадратный корень из 25)

- round() – значение, округленное до ближайшего целого. Если первое число после запятой от 0 до 4, то округление производится к меньшему по модулю целому, если же первое число после запятой от 5 до 9, то округление производится к большему. Например:

var x = 2.499;

var y = 2.5;

document.write(Math.round(x)); // округлено до 2

document.write(Math.round(y)); // округлено до 3

- ceil() – значение, округленное до ближайшего большего целого. Например:

var x = 2.499;

var y = 2.5;

document.write(Math.ceil(x)); // округлено до 3

document.write(Math.ceil(y)); // округлено до 3

- floor() – значение, округленное до ближайшего меньшего целого. Например:

var x = 2.499;

var y = 2.5;

document.write(Math.floor(x)); // округлено до 2

document.write(Math.floor(y)); // округлено до 2

- max(<Список чисел через запятую>) – максимальное значение из списка. Например: *document.write(Math.max(3, 10, 6)); // 10*
- min(<Список чисел через запятую>) – минимальное значение из списка. Например: *document.write(Math.min(3, 10, 6)); // 3*
- random() – случайное число от 0 до 1. Например:

document.write(Math.random()); // например, 0.9778613566886634

Для того чтобы получить случайное целое число от 0 до 9, нужно возвращаемое методом random() значение умножить на 9,9999, а затем округлить число до ближайшего меньшего целого при помощи метода floor(). Например:

*var x = Math.floor(Math.random()*9.9999);*

document.write(x);

Если несколько раз обновить веб-страницу, то можно увидеть, что число будет изменяться случайным образом в пределах от 0 до 9 включительно. Это может понадобиться, например, в случае если имеется четыре баннера 468.60, которые необходимо выводить на странице случайным образом:

*var x = Math.floor(Math.random()*3.9999);*

document.write('<img src="banner' + x + '.gif" width="480" height="60">');

Четыре баннера с именами banner0.gif, banner1.gif, banner2.gif и banner3.gif должны быть расположены в одной папке с файлом, в котором находится исполняемый скрипт.

Названия файлов с баннерами можно сделать произвольными, добавив их в массив. Например:

```
var Mass = [ "banner-red.gif", "banner-blue.jpeg",
"banner-gray.gif", "banner-white.png" ];
var x = Math.floor(Math.random()*3.9999);
document.write('<img src="' + Mass[x]);
document.write('" width="480" height="60">');
```

8.9. Класс Date. Получение текущей даты и времени

Класс Date позволяет работать с датой и временем. Экземпляры класса создаются таким образом:

```
<Экземпляр класса> = new Date();
<Экземпляр класса> = new Date(<Количество миллисекунд>);
<Экземпляр класса> = new Date(<Год>, <Месяц>, <День>, <Часы>,
<Минуты>, <Секунды>, <Миллисекунды>);
```

Класс Date поддерживает следующие методы:

- toString() преобразует дату в строку и возвращает ее. Например:

```
var d = new Date();
document.write(d.toString());
// В Opera: Fri, 30 Oct 2012 01:07:17 GMT+0300
// В Firefox: Fri Oct 30 2012 01:07:17 GMT+0300
// В IE: Fri Oct 30 01:07:17 UTC+0300 2012
```

- toLocaleString() преобразует дату в строку, используя интернациональные установки системы, и возвращает ее. Например:

```
var d = new Date();
document.write(d.toLocaleString());
// В Opera: 30.10.2012 1:11:27
// В Firefox: 30 Октябрь 2012 г. 1:11:27
// В IE: 30 октября 2012 г. 1:11:27
```

▪ valueOf() позволяет определить количество миллисекунд, прошедших с 01.01.1970 00:00:00:

```
var d = new Date();
document.write(d.valueOf()); // 1256854444062
```

▪ getDate() возвращает день месяца (от 1 до 31). Например:

```
var d = new Date();
document.write(d.getDate());// 30
```

▪ getDay() дает возможность узнать день недели (от 0 для воскресенья до 6 для субботы). Например:

```
var Mass = [ "воскресенье", "понедельник", "вторник", "среда", "четверг",
"пятница", "суббота" ];
var d = new Date();
document.write(Mass[d.getDay()]); // пятница
```

▪ getMonth() возвращает месяц (от 0 для января до 11 – для декабря):

```
var Mass = [ "январь", "февраль", "март", "апрель", "май", "июнь",
"июль", "август", "сентябрь", "октябрь", "ноябрь", "декабрь" ];
var d = new Date();
document.write(Mass[d.getMonth()]); // октябрь
```

Для получения номера текущего месяца к возвращаемому значению необходимо прибавить единицу. Например:

```
var d = new Date();
var Month = d.getMonth() + 1;
document.write(Month); // 10
```

▪ getFullYear() позволяет определить год. Например:

```
var d = new Date();
document.write(d.getFullYear()); // 2012
```

▪ getHours() возвращает час (от 0 до 23). Например:

```
var d = new Date();
document.write(d.getHours()); // 1
```

▪ getMinutes() позволяет получить минуты (от 0 до 59). Например:

```
var d = new Date();
document.write(d.getMinutes()); // 23
```

- getSeconds() возвращает секунды (от 0 до 59). Например:

```
var d = new Date();
document.write(d.getSeconds()); // 20
```

- getMilliseconds() возвращает миллисекунды (от 0 до 999). Например:

```
var d = new Date();
document.write(d.getMilliseconds()); // 156
```

- getTime() позволяет определить количество миллисекунд, прошедших с 01.01.1970 00:00:00. Например:

```
var d = new Date();
document.write(d.getTime()); // 1256855182843
```

Рассмотрим работу с датой и временем на конкретном примере.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Текущая дата и время</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<script type="text/javascript">
function f_Date(Str) { Str += ""; // Преобразуем число в строку
if (Str.length==1) return ("0" + Str); else return Str; }
function f_Year(Year) { Year += ""; // Преобразуем число в строку
return Year.substr(2); }
</script>
</head>
<body>
<script type="text/javascript">
var d = new Date();
var msg;
var Day = [ "воскресенье", "понедельник", "вторник", "среда", "четверг",
"пятница", "суббота" ];
var Month = [ "января", "февраля", "марта", "апреля", "мая", "июня",
"июля", "августа", "сентября", "октября", "ноября", "декабря" ];
```

```
msg = "Сегодня <br>" + Day[d.getDay()] + " ";
msg += d.getDate() + " ";
msg += Month[d.getMonth()] + " ";
msg += d.getFullYear() + " ";
msg += f_Date(d.getHours()) + ":";
msg += f_Date(d.getMinutes()) + ":";
msg += f_Date(d.getSeconds()) + "<br>";
msg += f_Date(d.getDate()) + ".";
msg += f_Date(d.getMonth() + 1) + ".";
msg += f_Year(d.getFullYear());
document.write(msg);
</script>
</body>
</html>
```

В окне веб-браузера отобразится надпись:

Сегодня
пятница 30 октября 2012 01:36:29
30.10.12

Надпись будет меняться, поскольку работа осуществляется с текущими датой и временем.

В примере были использованы две специально созданные функции:

- f_Date(Str) – если параметр состоит из одной цифры, то функция добавляет перед ним 0 и возвращает строку. Если не применить функцию, то, например, дата 05.04.2012 будет выглядеть 5.4.2012, так как методы класса Date возвращают число;
- f_Year(Year) – функция возвращает последние две цифры года.

8.10. Класс Function (функции)

Класс Function позволяет использовать функцию как экземпляр класса. Делается это таким образом:

<Имя функции> = new Function(<Параметр1>, ..., <ПараметрN>, <Тело функции>);

Например, функцию суммирования двух чисел

function f_Sum(x, y) { return x + y; }

можно переписать так:

var f_Sum = new Function ("x", "y", "return x + y");

Однако, указывать "тело" функции в виде строки очень неудобно. Поэтому данный способ практически не используется.

Вместо этого применяются анонимные функции. Например:

var f_Sum = function(x, y) { return x + y; };

Вызывать функцию можно, например, таким способом:

document.write(f_Sum(5, 6)); // 11

При использовании анонимных функций следует учитывать, что при указании внутри функции глобальной переменной будет сохранена ссылка на эту переменную, а не на ее значение. Например:

```
var x = 5;
var f_Sum = function() {
return x; } // Сохраняется ссылка, а не значение переменной x !
document.write(f_Sum()); // 5
x = 10; // Изменили значение
document.write(f_Sum()); // 10, а не 5
```

8.11. Класс Arguments. Функции с произвольным количеством аргументов

Класс массива Arguments позволяет получить доступ ко всем аргументам, переданным функции. Массив доступен только внутри тела функции. Получить доступ к аргументу можно, указав его индекс, а свойство length позволяет определить количество аргументов, переданных функции. Например:

```
function f_Sum(x, y) {
return arguments[0]+arguments[1]; }
document.write(f_Sum(5, 6)); // 11
```

При использовании массива аргументов можно передать функции больше аргументов, чем первоначально объявлено.

Например, можно просуммировать сразу несколько чисел, а не только два:

```
function f_Sum(x, y) { var z = 0;
for (var i=0, c=arguments.length; i<c; i++) {
z += arguments[i]; }
return z; }
document.write(f_Sum(5, 6, 7, 20)); // 38
```

8.12. Класс RegExp. Проверка значений с помощью регулярных выражений

Класс RegExp позволяет осуществить поиск в строке с помощью регулярных выражений, т.е. шаблонов для поиска определенных комбинаций метасимволов. Регулярные выражения позволяют осуществлять очень сложный поиск.

Создать экземпляр класса RegExp можно двумя способами:

1. <Экземпляр класса> = new RegExp(<Регулярное выражение> [<Модификатор>]).

2. <Экземпляр класса> = /<Регулярное выражение>/[<Модификатор>].

Необязательный параметр <Модификатор> задает дополнительные параметры поиска. Он может содержать следующие символы:

- i – поиск без учета регистра;
- g – глобальный поиск (поиск в строке всех вхождений регулярного выражения);
- m – многострочный режим. Символ ^ соответствует началу каждой подстроки, а $ – концу каждой подстроки. Например:

```
var p = new RegExp("^[0-9]$", "mg");
var Str = "1\n2\n3\nстрока\n4";
Mass = Str.match(p);
document.write(Mass.join(", ")); // Выведет: 1, 2, 3, 4
```

- gi – глобальный поиск без учета регистра символов.

Методы search(), match() и replace() можно использовать таким образом:

- search(<Регулярное выражение>) возвращает номер позиции первого

вхождения подстроки, совпадающей с регулярным выражением. Например:

```
var p = new RegExp("200[14]");
var Str = "2000, 2001, 2002, 2003, 2004";
document.write(Str.search(p)); // 6
```

Шаблону 200[14] соответствуют только два года: 2001 и 2004.

- match(<Регулярное выражение>) возвращает массив с результатами поиска, совпадающими с регулярным выражением. Например:

```
var p = new RegExp("200[14]");
var Str = "2000, 2001, 2002, 2003, 2004";
var Mass = [];
Mass = Str.match(p);
for (var i=0, c=Mass.length; i<c; i++)
document.write(Mass[i] + "<br>");
```

Этот пример выведет только 2001, так как не указан модификатор глобального поиска g. Чтобы получить все вхождения необходимо его модифицировать следующим образом:

```
var p = new RegExp("200[14]", "g");
var Str = "2000, 2001, 2002, 2003, 2004";
var Mass = [];
Mass = Str.match(p);
for (var i=0, c=Mass.length; i<c; i++)
document.write(Mass[i] + "<br>");
```

Теперь будут выведены все подстроки, совпадающие с регулярным выражением: и 2001, и 2004.

- replace(<Регулярное выражение>, <Текст для замены>) возвращает строку, которая является результатом поиска и замены в исходной строке с использованием регулярного выражения. Например:

```
var p = new RegExp("200[14]", "g");
var Str = "2000, 2001, 2002, 2003, 2004";
Str = Str.replace(p, "2007");
document.write(Str); // "2000, 2007, 2002, 2003, 2007"
```

В качестве второго параметра можно также указать ссылку на функ-

цию. Через первый параметр в функции доступна строка, полностью соответствующая шаблону. Через остальные параметры доступны подвыражения, которые соответствуют фрагментам, заключенным в шаблоне в круглые скобки. Ниже представлен пример нахождения всех чисел в строке с последующим прибавлением к ним числа 10.

```
var p = new RegExp("[0-9]([0-9]+)", "g");
var Str = "2000, 2001, 2002, 2003, 2004";
Str = Str.replace(p, function(s, x) {
document.write(x + ", ");
var n = parseInt(s);
n += 10;
return n + ""; });
document.write("<br>" + Str);
// "000, 001, 002, 003, 004, "
// "2010, 2011, 2012, 2013, 2014"
```

В строке для замены можно использовать специальные переменные $1, ..., $N, через которые доступны фрагменты, заключенные в шаблоне в круглые скобки. Ниже представлен пример кода, с помощью которого можно поменять местами два тега:

```
var p = new RegExp("<([a-z]+)><([a-z]+)>");
var Str = "<br><hr>";
Str = Str.replace(p, "&lt;$2&gt;&lt;$1&gt;");
document.write(Str);
// Выведет в окне веб-браузера: "<hr><br>"
```

Метод split(<Регулярное выражение>, [<Лимит>]) также поддерживает регулярные выражения. Возвращает массив, полученный в результате разделения строки на подстроки по фрагменту, соответствующему регулярному выражению. Если второй параметр присутствует, то он задает максимальное количество элементов в результирующем массиве. Например:

```
var Str = "1 2 3\n4\t5\r6";
var Mass = Str.split(/\s/);
document.write(Mass.join(", ")); // "1, 2, 3, 4, 5, 6"
```

```
var Mass2 = Str.split(/\s/, 3);
document.write(Mass2.join(", ")); // "1, 2, 3"
```

Вместо методов класса String можно воспользоваться методами класса RegExp:

- test(<Строка>) возвращает true или false в зависимости от того был поиск успешным или нет. Ниже представлен пример проверки правильности введенной даты.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Проверка вводимых данных</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<script type="text/javascript">
var d = window.prompt("Введите дату в формате день.месяц.год", "");
if (d==null) { document.write("Вы нажали Отмена"); }
else { var p = /^[0-3]\d\.[01]\d\.\d{4}$/;
if (p.test(d)) document.write("Дата введена правильно");
else document.write("Вы неправильно ввели дату"); }
</script>
</body>
</html>
```

- exec(<Строка>) позволяет получить массив с результатами поиска, совпадающими с регулярным выражением. Например:

```
var p = new RegExp("(\\d{2}):(\\d{2}):(\\d{2})");
var Str = "Sun Apr 29 18:47:27 UTC+0400 2007";
var Mass = [];
Mass = p.exec(Str);
document.write(Mass.join("<br>"));
```

В результате будет получен следующий результат:

18:47:27

18

47

27

Первая строка соответствует найденному фрагменту (элемент массива с индексом 0). Вторая, третья и четвертая строки содержат фрагменты, соответствующие группам метасимволов (\\d{2}), заключенных в круглые скобки. Номер скобок по порядку следования в регулярном выражении соответствует индексу фрагмента в массиве.

8.13. События

При взаимодействии пользователя с веб-страницей происходят события. События – это своего рода извещения системы о том, что пользователь выполнил какое-либо действие или внутри самой системы возникло некоторое условие.

События возникают при щелчке на элементе, перемещении мыши, нажатии клавиши на клавиатуре, изменении размеров окна, по окончанию загрузки веб-страницы и т.д.

Зная, какие события может генерировать тот или иной элемент веб-страницы, можно написать функцию для обработки этого события. Например, при отправке данных формы возникает событие onsubmit. При наступлении этого события можно проверить данные, введенные пользователем, и если они не соответствуют ожидаемым, прервать отправку данных. Все названия событий начинаются с префикса on.

События мыши

Основными событиями мыши являются:

• onmousedown возникает при нажатии кнопки мыши на элементе веб-страницы или самой странице;

• onmouseup возникает при отпускании ранее нажатой кнопки мыши;

• onclick возникает при щелчке мыши на элементе веб-страницы или на самой веб-странице;

• ondblclick возникает при двойном щелчке мыши;

• onmousemove возникает при любом перемещении мыши;

• onmouseover возникает при наведении курсора мыши на элемент веб-страницы;

• onmouseout возникает при выведении курсора мыши с элемента веб-страницы;

• onselectstart возникает, когда начинается выделение текста;

• onselect возникает при выделении элемента;

• oncontextmenu возникает при нажатии правой кнопки мыши для вывода контекстного меню.

События клавиатуры

Основными событиями клавиатуры являются:

• onkeydown возникает при нажатии клавиши на клавиатуре;

• onkeypress возникает аналогично событию onkeydown, но возвращает значение кода символа в кодировке Unicode. Возникает постоянно, пока пользователь не отпустит клавишу;

• onkeyup возникает при отпускании ранее нажатой клавиши клавиатуры;

• onhelp возникает при нажатии клавиши F1.

События документа

Основными событиями документа являются:

• onload возникает после загрузки веб-страницы;

• onscroll возникает при прокручивании содержимого элемента страницы, документа, окна или фрейма;

• onresize возникает при изменении размеров окна;

• onbeforeunload возникает перед выгрузкой документа;

• onunload возникает непосредственно перед выгрузкой документа. Наступает после события onbeforeunload;

• onbeforeprint возникает перед распечаткой документа или вывода его на предварительный просмотр;

• onafterprint возникает после распечатки документа или вывода его на предварительный просмотр.

События формы

Основными событиями формы являются:

- onsubmit – возникает при отправке данных формы;
- onreset – возникает при очистке формы;
- onblur – возникает при потере фокуса элементом формы;
- onchange – возникает при изменении данных в текстовом поле и перемещении фокуса на другой элемент формы либо при отправке данных формы (наступает перед событием onblur);
- onfocus – при получении фокуса элементом формы.

8.14. Последовательность событий

События возникают последовательно, например, последовательность событий при нажатии кнопкой мыши на элементе страницы может быть такой:

1. onmousedown
2. onmouseup
3. onclick

А при двойном нажатии последовательность может быть такой:

1. onmousedown
2. onmouseup
3. onclick
4. ondblclick

Это означает, что событие ondblclick возникает после события onclick.

При нажатии клавиши на клавиатуре последовательность может быть такой:

1. onkeydown
2. onkeypress
3. onkeyup

Ниже представлен пример кода, в котором наглядно представлена последовательность событий.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
```

```
<head>
<title>Последовательность событий</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<script type="text/javascript">
function f_print(Str) {
var div1 = document.getElementById("div1");
div1.innerHTML += Str + "<br>"; }
</script>
</head>
<body onload="f_print('Событие onload - Страница загружена');"
onmousedown="f_print('Событие onmousedown - Нажали');"
onmouseup="f_print('Событие onmouseup - Отпустили');"
onclick="f_print('Событие onclick - Щелчок');"
onkeydown="f_print('Событие onkeydown - Нажали');"
onkeypress="f_print('Событие onkeypress - Нажали');"
onkeyup="f_print('Событие onkeyup - Отпустили');">
<p onmouseover="f_print('Событие onmouseover - Навели курсор');"
onmouseout="f_print('Событие onmouseout - Убрали курсор');">
Щелкните мышью в любом месте страницы
</p><p></p>
<div id="div1"></div>
</body>
</html>
```

Собятия в примере будут возникать в такой последовательности. После загрузки возникнет событие onload. Если щелкнуть в любом месте окна и не отпускать кнопку мыши, возникнет только событие onmousedown. Если отпустить, то возникают сразу два события: onmouseup и onclick. Если нажать любую клавишу клавиатуры и не отпускать, то возникнут сразу два события: onkeydown и onkeypress. Причем если продолжать удерживать клавишу нажатой, то событие onkeypress будет повторяться. Если отпустить, то возникнет событие onkeyup. Если навести курсор мыши на надпись "Щелкните мышью в любом месте стра-

ницы", то возникнет событие onmouseover. Если убрать курсор с надписи, то возникнет событие onmouseout.

8.15. Реализация обработчиков событий

Обработчики событий можно использовать как атрибуты тегов. Например:

```
<span style="color: red" onclick="f_print('Событие onclick - SPAN', event);">
здесь</span>
```

Но это не единственный вариант написания обработчиков. Написать обработчик можно с помощью параметров for и event тега <script>. Для этого элемент веб-страницы должен иметь параметр id. В параметре for указывается id элемента страницы, для которого создается обработчик, а в параметре event указывается обрабатываемое событие. Например:

```
<html>
<head>
<title>Обработчик события</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<script type="text/javascript" for="txt" event="onclick">
window.alert('Вы кликнули на слове "здесь"');
</script>
</head>
<body>
<p>Щелкните мышью
<span style="color: red" id="txt">здесь</span>
</p>
</body>
</html>
```

Можно назначить обработчик с помощью указателя функции. При этом имя функции обязательно должно быть указано без скобок и дополнительных атрибутов. Например:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

```
<html>
<head>
<title>Обработчик события</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<script type="text/javascript">
function f_click(e) {
e = e || window.event;
window.alert('Вы кликнули на слове "здесь"');
// this – это ссылка на элемент, вызвавший событие
this.innerHTML = "новый текст";
// Прерывание всплывания событий
if (e.stopPropagation) e.stopPropagation();
else e.cancelBubble = true; } // Для IE
</script>
</head>
<body>
<p onclick="window.alert('Событие onclick - Абзац');">
Щелкните мышью
<span style="color: red" id="txt">здесь</span>
</p>
<script type="text/javascript">
// Обратите внимание: название функции указывается без скобок
document.getElementById("txt").onclick = f_click;
</script>
</body>
</html>
```

Кроме того, обработчик можно написать, используя анонимную функцию. Например:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
```

```
<title>Обработчик события</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<p onclick="window.alert('Событие onclick - Абзац');">
Щелкните мышью
<span style="color: red" id="txt">здесь</span>
</p>
<script type="text/javascript">
// Использование анонимной функции
document.getElementById("txt").onclick = function(e) {
e = e || window.event;
window.alert('Вы кликнули на слове "здесь"');
// this – это ссылка на элемент, вызвавший событие
this.innerHTML = "новый текст";
// Прерывание всплывания событий
if (e.stopPropagation) e.stopPropagation();
else e.cancelBubble = true; }
</script>
</body>
</html>
```

Назначить обработчик события в модели DOM Level 2 позволяет метод addEventListener(), который имеет такой формат:

<Элемент>.addEventListener(<Событие>, <Ссылка на функцию>, <Перехват>);

Удалить обработчик события можно с помощью метода removeEventListener(), который имеет такой формат:

<Элемент>.removeEventListener(<Событие>, <Ссылка на функцию>, <Перехват>);

В параметре <Событие> указывается название события без префикса "on" (например, click вместо onclick). Ссылка на функцию-обработчик указывается во втором параметре. В эту функцию в качестве аргумента переда-

ется ссылка на объект event, а внутри функции через ключевое слово this доступна ссылка на текущий элемент. Если в параметре <Перехват> указать значение true, то событие будет перехватываться на этапе всплывания от вложенных элементов, а если false – то обрабатывается событие самого элемента. Ниже представлен пример использования этого параметра:

```
<div><span id="span1">span1
<span id="span2"> Щелкните здесь (span2) </span></span>
</div>
<script type="text/javascript">
function f_click(e) { // e - ссылка на объект event
window.alert("Элемент " + this.getAttribute("id") +
". Событие возникло в " + e.target.getAttribute("id")); }
if (document.addEventListener) { // В IE не работает
var span1 = document.getElementById("span1");
var span2 = document.getElementById("span2");
span1.addEventListener("click", f_click, true);
span2.addEventListener("click", f_click, false); }
</script>
```

При щелчке на фразе "Щелкните здесь" возникнет последовательность событий:

Элемент span1. Событие возникло в span2
Элемент span2. Событие возникло в span2

Таким образом, событие, возникшее во вложенном элементе, вначале обрабатывается элементом-родителем, а затем самим элементом. Если заменить true на false, то последовательность будет другой:

Элемент span2. Событие возникло в span2
Элемент span1. Событие возникло в span2

Ниже представлен пример назначения обработчика для всех кнопок (type="button"), а также реализации обработчика, отрабатывающего только один раз:

```
<input type="button" id="btn1" value="Кнопка 1">
<input type="button" id="btn2" value="Кнопка 2">
```

```
<script type="text/javascript">
function f_click1(e) { // e - ссылка на объект event
// Сработает при каждом щелчке на любой кнопке
window.alert("Обработчик 1. Кнопка " + e.target.getAttribute("id")); }
function f_click2() { // Сработает только 1 раз
window.alert("Обработчик 2");
// Удаление обработчика
// this – ссылка на текущий элемент
this.removeEventListener("click", f_click2, false); }
if (document.addEventListener) { // В IE не работает
var tags = document.getElementsByTagName("input");
for (var i=0, len=tags.length; i<len; i++) {
if (tags[i].type=="button")
tags[i].addEventListener("click", f_click1, false); }
var elem = document.getElementById("btn1");
elem.addEventListener("click", f_click2, false); }
</script>
```

Веб-браузер Internet Explorer не поддерживает методы addEventListener() и removeEventListener(). Для назначения обработчика в этом веб-браузере, начиная с пятой версии, предназначен метод attachEvent(), который имеет следующий формат:

<Элемент>.attachEvent(<Событие>, <Ссылка на функцию>);

Удалить обработчик события можно с помощью метода detachEvent(), который имеет следующий формат:

<Элемент>.detachEvent(<Событие>, <Ссылка на функцию);

В параметре <Событие> указывается название события с префиксом "on" (например, onclick). Ссылка на функцию-обработчика указывается во втором параметре. В эту функцию в качестве аргумента передается ссылка на объект event. Внутри функции ключевое слово this ссылается на объект window, а не на текущий элемент.

Ниже представлен пример использования методов attachEvent() и detachEvent() для назначения и удаления обработчиков.

```
<input type="button" id="btn1" value="Кнопка 1">
<input type="button" id="btn2" value="Кнопка 2">
<script type="text/javascript">
function f_click1(e) { // Сработает при каждом щелчке на любой кнопке
window.alert("Обработчик 1. Кнопка " + e.srcElement.id); }
function f_click2() { // Сработает только 1 раз
window.alert("Обработчик 2");
// Удаление обработчика
var elem = document.getElementById("btn1");
elem.detachEvent("onclick", f_click2); }
if (document.attachEvent) { // Работает в IE 5+, Opera 9.02
var tags = document.getElementsByTagName("input");
for (var i=0, len=tags.length; i<len; i++) { if (tags[i].type=="button")
tags[i].attachEvent("onclick", f_click1); }
var elem = document.getElementById("btn1");
elem.attachEvent("onclick", f_click2); }
</script>
```

До пятой версии в Internet Explorer можно назначать обработчики только как параметры тегов или присваиванием ссылки на функцию свойству обработчика элемента документа. В этом случае объект event не передается в качестве параметра. Вместо него следует использовать глобальное свойство event объекта window.

Ниже представлен пример кроссбраузерного варианта назначения обработчика для события onload.

```
function f_load(e) {
var e = e || window.event; // Объект event
window.alert("Событие onload"); }
if (window.addEventListener) { // DOM Level 2
window.addEventListener("load", f_load, false); }
else if (window.attachEvent) { // IE 5+
window.attachEvent("onload", f_load); }
else window.onload = f_load; // IE 4-
```

8.16. Объект event. Вывод координат курсора и кода нажатой клавиши. Вывод сообщений при нажатии комбинации клавиш

Объект event позволяет получить детальную информацию о произошедшем событии и выполнить необходимые действия. Объект event доступен только в обработчиках событий. При наступлении следующего события все предыдущие значения свойств сбрасываются.

Объект event имеет следующие свойства:

- srcElement – ссылка на элемент, который является источником события. В модели DOM Level 2 используется свойство target;
- currentTarget – в модели DOM Level 2 возвращает ссылку на элемент, в котором обрабатывается событие. Ссылается на тот же элемент, что и ключевое слово this внутри обработчика события. Значение свойства currentTarget может не совпадать со значением свойства target;
- type – строка, содержащая тип события. Возвращается в нижнем регистре и без префикса on. Например, при событии onclick свойство type равно click;
- clientX и clientY – координаты события по осям X и Y в клиентских координатах;
- screenX и screenY – координаты события по осям X и Y относительно окна;
- offsetX и offsetY – координаты события по осям X и Y относительно контейнера;
- x и y –координаты события по осям X и Y. В модели DOM Level 2 этих свойств нет;
- button – число, указывающее нажатую кнопку мыши. Может принимать следующие значения:
  - 0 – кнопки не были нажаты;
  - 1 – нажата левая кнопка мыши;
  - 2 – нажата правая кнопка мыши;
  - 3 – левая и правая кнопки мыши были нажаты одновременно;
  - 4 – нажата средняя кнопка.

В модели DOM Level 2 значения будут другими:

- 0 – нажата левая кнопка мыши;
- 1 – нажата средняя кнопка;
- 2 – нажата правая кнопка мыши.

▪ keyCode – код нажатой клавиши клавиатуры. В веб-браузере Firefox при нажатии обычной клавиши в обработчике события onkeypress свойство keyCode имеет значение 0, а код символа доступен через свойство charCode. Если нажата только функциональная клавиша, то свойство charCode имеет значение 0, а код символа доступен через свойство keyCode;

▪ altKey возвращает true, если в момент события была нажата клавиша Alt;

▪ altLeft возвращает true, если была нажата левая клавиша Alt, и false, если правая. В модели DOM Level 2 этого свойства нет;

▪ ctrlKey возвращает true, если была нажата клавиша Ctrl;

▪ ctrlLeft возвращает true, если была нажата левая клавиша Ctrl, и false, если правая. В модели DOM Level 2 этого свойства нет;

▪ shiftKey возвращает true, если была нажата клавиша Shift;

▪ shiftLeft возвращает true, если была нажата левая клавиша Shift, и false, если правая. В модели DOM Level 2 этого свойства нет;

▪ cancelBubble указывает, будет ли событие передаваться по иерархии объектов или нет. Для прерывания всплывания событий необходимо этому свойству присвоить значение true. В модели DOM Level 2 используется метод stopPropagation();

▪ returnValue задает будет ли выполняться действие по умолчанию для элемента страницы. Для прерывания действия по умолчанию необходимо этому свойству присвоить значение false (см. пример ниже). В модели DOM Level 2 используется метод preventDefault().

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Прерывание действия по умолчанию</title>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<script type="text/javascript">
function f_print(Str, e) {
window.alert(Str);
e = e || window.event;
if (e.preventDefault) e.preventDefault();
else e.returnValue = false; }
</script>
</head>
<body>
<p>
<a href="file.html" onclick="f_print('Перехода по ссылке не будет!', event);">
Нажмите для перехода по ссылке</a><br><br>
<a href="file.html" onclick="window.alert('Перехода по ссылке не будет!');
return false;">Нажмите для перехода по ссылке</a></p>
</body>
</html>
```

В этом примере рассмотрены два метода прерывания действия по умолчанию. В первой ссылке прерывание действия по умолчанию реализовано с помощью свойства returnValue объекта event. Во второй ссылке прерывание осуществляется возвратом значения false;

- fromElement – ссылка на элемент, с которого переместился курсор мыши. В модели DOM Level 2 используется свойство relatedTarget;
- toElement – ссылка на элемент, на который пользователь перемещает курсор мыши. В модели DOM Level 2 используется свойство relatedTarget;
- repeat – true, если событие onkeypress наступило повторно в результате удержания клавиши нажатой. В модели DOM Level 2 этого свойства нет;
- propertyName – имя атрибута тега, стиля или свойства элемента страницы, значение которого изменилось. В модели DOM Level 2 этого свойства нет.

С помощью свойств объекта event можно вывести координаты курсо-

ра и код нажатой клавиши. Например:

```
<html>
<head>
<title>Координаты курсора и код нажатой клавиши</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<script type="text/javascript">
function f_unload() {
event.returnValue = "Хотите покинуть документ?"; }
function f_body() {
switch (event.type) {
case "mousemove":
var m1, m2;
m1 = event.clientX;
m2 = event.clientY;
var div1 = document.getElementById("div1");
div1.innerHTML = "clientX, clientY: x - " + m1 + " y - " + m2;
m1 = event.screenX;
m2 = event.screenY;
var div2 = document.getElementById("div2");
div2.innerHTML = "screenX, screenY: x - " + m1 + " y - " + m2;
m1 = event.offsetX;
m2 = event.offsetY;
var div3 = document.getElementById("div3");
var div4 = document.getElementById("div4");
var div5 = document.getElementById("div5");
div3.innerHTML = "offsetX, offsetY: x - " + m1 + " y - " + m2;
div4.innerHTML = "x, y: x - " + event.x + " y - " + event.y;
div5.innerHTML = "Тег: " + event.srcElement.tagName;
break;
case "keypress":
var div6 = document.getElementById("div6");
div6.innerHTML = "код нажатой клавиши - " + event.keyCode;
```

```
if (event.ctrlLeft && event.keyCode==10) {
window.alert("Нажата левая клавиша Ctrl + Enter"); }
if (!event.ctrlLeft && event.keyCode==10) {
window.alert("Нажата правая клавиша Ctrl + Enter"); }
break;
case "contextmenu":
event.returnValue = false;
break;
case "selectstart":
event.returnValue = false;
break; } }
</script>
</head>
<body onkeypress="f_body();" onmousemove="f_body();"
onbeforeunload="f_unload();">
<p oncontextmenu="f_body();">
Над этим абзацем нельзя вывести контекстное меню</p>
<div id="div5"></div>
<div id="div1"></div>
<div id="div2"></div>
<div id="div3"></div>
<div id="div4"></div>
<div id="div6">Нажмите клавишу на клавиатуре, чтобы увидеть ее
код</div>
<p>Нажмите левую или правую клавишу Ctrl вместе с Enter, чтобы уви-
деть сообщение</p>
<p onselectstart="f_body();">Этот абзац нельзя выделить отдельно от
других абзацев</p>
<div><a href="file.html">Нажмите для перехода по ссылке</a></div>
</body>
</html>
```

Практические задания

Вариант 1

1. Создайте кнопку с надписью "Нажата 0 раз". При нажатии на кнопку, надпись на ней должна изменяться на "Нажата 1 раз", т.е. кнопка должна отображать количество нажатий на неё.

2. Реализуйте с помощью функции вывод в окно полной информации о текущей дате и времени. Например, "14 мая 2012 года, вторник, 2:53:44pm".

Вариант 2

1. Выведите три одинаковых изображения. При наведении мыши на первое изображение должны изменяться второе и третье, при наведении мыши на второе изображение должны изменяться третье и первое, при наведении мыши на третье изображение, оно должно заменяться на четвертое изображение. После схода мыши с любого изображения все они должны вернуться в исходное состояние.

2. Реализуйте с помощью функции вывод в окно количества дней, прошедших со дня Вашего рождения.

Вариант 3

1. Реализуйте с помощью функции изменение цвета заголовка с черного на оранжевый при наведении на него указателя мыши. Создайте три кнопки с названиями цветов: красный, зеленый, синий, при нажатии на которые цвет заголовка будет изменяться в соответсвии с названием кнопки.

2. Реализуйте с помощью функции вывод в окно количества часов и минут, прошедших с начала суток.

Вариант 4

1. Реализуйте движение произвольного изображения из левого верхнего угла веб-страницы в ее правый нижний угол.

2. Реализуйте с помощью функции вывод в окно сообщения, определяющего является ли текущий год високосным.

Вариант 5

1. Реализуйте движение произвольного изображения по периметру экрана, т.е. из левого верхнего угла в правый верхний угол, затем в правый нижний угол, далее в левый нижний угол и обратно в левый верхний угол.

2. Реализуйте с помощью функции вывод в окно количества минут, прошедших с начала пары.

Контрольные вопросы

1. Как создать одномерный массив?
2. Как присвоить значение элементам одномерного массива?
3. Назовите свойство, в котором хранится размер массива.
4. Как передать из функции несколько переменных-массивов?
5. Как создать двумерный массив?
6. Назовите тип и значение переменной, возникающей при делении на ноль.
7. Назовите тип и значение переменной, которой не присвоено никакое значение.
8. Как реализовать присвоение значения переменной с использованием метода prompt()?
9. Как создаются константы и чем они отличаются от переменных?
10. В чем различия вывода данных с помощью метода alert() и с помощью свойства innerHTML?
11. В чем различия вывода данных с помощью метода alert() и с помощью метода document.write()?
12. Какая функция или какой метод используются для округления вещественного числа до целого?
13. Для чего используется объект event?
14. Для чего используются регулярные выражения?
15. Какие методы существуют в классе Function?
16. Какие методы существуют в классе Date?
17. Какие методы существуют в классе Math?

18. Какие методы существуют в классе String?
19. Какие методы существуют в классе Number?
20. Какие методы существуют в классе Global?
21. Для чего предназначен ассоциативный массив? Как его создать?
22. Как изменить свойства документов при помоши языка JavaScript?

КРАТКИЙ СЛОВАРЬ ТЕРМИНОВ ИЗ ОБЛАСТИ ВЕБ-ТЕХНОЛОГИЙ

АКЦЕПТОР – страница, на которую сослались, т.е. поставили ссылку.

АНКОР – текст ссылки на определенный ресурс в сети Интернет.

АНКОР ЛИСТ – список всех анкоров.

БАН ЛИСТ – «черный» список ресурсов сети Интернет, которые в силу разных причин не рассматриваются поисковыми системами.

БАЗЫ ДАННЫХ – совокупность технических данных о сайте в виде обобщенных таблиц, объединенных в одну большую базу. Если речь идет о MySql базах, то она часто применяется в сайтостроениии и может содержать в себе статьи сайта, комментарии и т.д. Это также позволяет восстанавить сайт с нуля, если предварительно была сделана резервная копия базы данных.

БРАУЗЕР – программа, с помощью которой пользователь просматривает веб-узлы в открытом доступе в сети Интернет.

БИРЖА ССЫЛОК – проект, который позволяет вебмастерам (создателям сайтов), продавать места на своих сайтах под ссылки рекламодателей (оптимизаторов), а оптимизаторам в свою очередь покупать эти места под ссылки.

БЭКАП (BACKUP) – это резервное копирование данных с целью их последующего восстановления в случае их повреждения или разрушения.

ВЕБ-УЗЕЛ – это ресурс сети Интернет с уникальным адресом, однозначно индентифицирующим данный ресурс.

ВЕБ-САЙТ – совокупность веб-страниц, расположенных в рамках одного домена, на одном дисковом пространстве.

ВИРТУАЛЬНЫЙ СЕРВЕР – дисковое пространство для сайтов, обычный хостинг с условиями для неглобальных проектов. Большинство блогов, сайтов, даже порталов размещены именно на таком хостинге.

ВИРТУАЛЬНЫЙ ВЫДЕЛЕННЫЙ СЕРВЕР – дисковое пространство (как правило, намного больше, чем у простого виртуального сервера), обеспеченное более мощным оборудованием, большими возможностями, поддержкой программного обеспечения и т.д., то есть

более гибкими параметрами для заказчика. Такие сервера арендуются для размещения мощных проектов, например глобальных Интернет-магазинов, сервисов социальных сетей и т.д.

ВНЕШНЯЯ ПОИСКОВАЯ ОПТИМИЗАЦИЯ – это внешнее продвижение сайта, то есть наращивание внешней ссылочной массы (количества ссылок с прочих ресурсов на данный ресурс).

ВНУТРЕННЯЯ ПОИСКОВАЯ ОПТИМИЗАЦИЯ – обработка материала на сайте самым оптимальным образом под поисковые системы (написание статей под поисковые запросы, включение ключевых слов в текст статей и т.д.).

ВЧ-СЧ-НЧ (ВЫСОКОЧАСТОТНЫЙ ЗАПРОС – СРЕДНЕЧАСТОТНЫЙ ЗАПРОС – НИЗКОЧАСТОТНЫЙ ЗАПРОС) – запросы, которые пользователи делают к поисковой системе с высокой, средней и низкой частотой соответственно. Частотность запроса зависит от его тематики.

ГИПЕРЛИНК (ССЫЛКА) – ссылка на конкретный ресурс в сети Интернет.

ДОМЕН – это уникальный адрес веб-сайта, который вводится в адресной строке браузера.

ДОМЕННОЕ ИМЯ – это уникальное название домена.

ДОМЕННАЯ ЗОНА – зона, в которой расположен домен. Доменная зона обуславливает принадлежность доменов к определенной территории, определенному региону, например, UA – Украина, либо к определенной деятельности, например, com – коммерческие сайты.

ДОРВЕЙ – сайт, предназначенный для раскрутки основного сайта. Как правило, дорвей содержит несколько страниц с большим количеством ключевых слов, ориентированных на запросы пользователей. Когда пользователь вводит запрос в поисковике, ему выдается список сайтов. Если пользователь заходит на сайт из этого списка, а этот сайт в свою очередь является дорвеем, то пользователь автоматически перебрасывается на основной сайт, при этом поисковый робот попасть на основной сайт не может.

ДОНОР – веб-страница или веб-сайт, которая размещает ссылку на другой ресурс. Благодаря этому вес того ресурса, на который сосла-

лись, увеличивается.

ЗЕРКАЛО – сайт, который является копией другого сайта, по мнению поисковой системы.

ИНДЕКС – совокупность данных поисковой машины, то есть база, в которую заносятся адреса страниц сайтов.

ИНДЕКСИРОВАНИЕ – процесс добавления страниц сайта в индекс поисковой системы поисковым роботом.

КАПЧА (CAPTHA) – защита от автоматического отправления данных при регистрациях на сайте, заполнении каких-либо полей (например, пароля и логина) и т.п. Создана с целью исключить возможность регистрации и других действий роботами-программами. Как правило, это цифры и/или буквы, которые нужно ввести в определенное поле, чтобы продолжить какое-либо действие.

КАСКАД ФИЛЬТРОВ – совокупность фильтров, которая применяется в том или ином случае.

КЛЮЧЕВЫЕ СЛОВА – это слова и словосочетания (реже предложения), используемые в запросах пользователей к поисковым системам. Список ключевых слов является семантическим ядром сайта.

КОНВЕРСИЯ – отношение количества посетителей сайта, которые выполнили определенные действия (например, кликнули по ссылке в блоке с контекстной рекламой), к общему количеству посетителей.

КОНТЕНТ – это содержимое сайта, вся информация, содержащаяся на сайте: изображения, видео, текст и т.д.

КОНТЕКСТНАЯ РЕКЛАМА – реклама в виде блока ссылок, которые отсылают к рекламируемому ресурсу.

КУКИ (COOKIES) – текстовый файл, в котором хранится информация о пользователе: его запросах, его учетной записи на каком-либо сайте и т.п.

КОПИПАСТ – способ добычи контента, при котором все статьи и материалы одного сайта копируются и без изменений размещаются на другом сайте. По сути дела, это плагиат и воровство контента и таким образом наполнять сайт нецелесообразно, поскольку поисковые системы не принимают такие сайты, более того, они накладывают различного рода санкции на такие сайты, вплоть до занесения в бан-лист.

КОПИРАЙТИНГ – это наполнение сайта уникальным контентом, авторскими материалами (копирайтер – писатель). Копирайтинг также может рассматриваться как отдельный способ заработка.

КОРНЕВОЙ КАТАЛОГ – основная папка первого уровня вложенности.

КРОСПОСТИНГ – массовое добавление нового контента на сайте на различные Интернет-ресурсы. Как правило, осуществляется автоматически с помощью различных онлайн сервисов или программ.

ЛОГ – это файл, создаваемый сервером. В данном файле хранится информация о степени активности на сайте: посещениях, числе проиндексированных страниц и т.д.

МЕРТВАЯ ССЫЛКА – ссылка, ведущая на несуществующую страницу.

МЕТАТЕГИ – служебные теги, которые сообщают браузеру или поисковому боту информацию о сайте. Например, ключевые слова статьи и т.д.

МОНЕТИЗАЦИЯ – получение прибыли от работы сайта, не предназначенного для заработка.

ПАРСЕР (ГРАББЕР) – программа или скрипт, которая копирует контент с других сайтов. Используется для быстрого наполнения сайта контентом.

ПЕРЕЛИНКОВКА – связывание страниц сайта с помощью ссылок на другие страницы этого же сайта. Ярким примером является сайт википедии, где на каждой странице есть ссылки на другие страницы, раскрывающие тот или иной термин на исходной странице.

ПЛАГИН – это модуль, который позволяет дополнять сайт какими-либо функциями, например, производить оплату прямо на сайте, реализовать слайд-шоу и т.д.

ПОСТИНГ – добавление контента на определенные ресурсы в сети Интернет: блоги, сайты, социальные закладки, сети и т.д.

ПОИСКОВАЯ ВЫДАЧА – список сайтов, которые выдает поисковая система в ответ на запрос пользователя. Поисковая выдача выстраивается согласно правилам ранжирования с учетом релевантности.

ПОИСКОВАЯ СИСТЕМА – это мощный Интернет-ресурс, предо-

ставляющий ответы на запросы пользователей в виде списка сайтов.

ПОИСКОВАЯ ОПТИМИЗАЦИЯ (search engine optimization, SEO) – оптимизация сайта под поисковые системы, т.е. совокупность действий, направленных на продвижение сайтов в поисковых системах.

ПРОТОКОЛ СОЕДИНЕНИЯ – протокол, который используется для соединения с веб-узлами в определенных целях. Например, для соединения с сайтом используется протокол http://, где после знака / (слэш), указывается адрес сайта, а для «заливки» сайта на хостинг используется протокол ftp://, где после знака / (слэш) указывается IP-адрес, идентифицирующий хостинг.

РАНЖИРОВАНИЕ – это формирование списка сайтов в ответ на определенный запрос пользователя.

РЕДИРЕКТ – перенаправление с одного сайта на другой или с одной страницы на другую.

РЕЛЕВАНТНОСТЬ – степень соответствия ответа запросу пользователя.

РЕССЕЛИНГ ХОСТИНГА – перепродажа хостинга.

РЕФЕРАЛ – пользователь, который зарегистрировался или воспользовался услугами какой-либо компании по рекомендации другого пользователя. При этом компания должна выдать пользователю, рекомендовавшему ее, уникальные промо-материалы, которые будут показывать, от какого конкретно пользователя «пришел» реферал. За это пользователь может получить часть прибыли от приобретенных рефералом услуг.

САТЕЛЛИТ – сайт, созданный для продвижения основного сайта.

СЕМАНТИЧЕСКОЕ ЯДРО – совокупность ключевых запросов по определенной тематике.

СГЕНЕРИРОВАННЫЙ ТЕКСТ – текст статей, который автоматически создается программой или скриптом.

СИМВОЛЬНЫЙ АДРЕС ДОМЕНА – адрес, соответствующий определенному домену и состоящий из латинских букв.

СКРИПТ – часть программного кода, которая содержит определенную последовательность действий для определенных обстоятельств.

СПАМ – массовая рассылка нежелательной корреспонденции,

например, многочисленных сообщений с назойливой рекламой по электронной почте. За это применяются различные санкции, как от поисковых систем, так и от других органов.

СИНОНИМАЙЗ (СИНОНИМАЙЗИНГ) – замена слов исходного текста синонимами.

СИНОНИМАЙЗЕР – программа или скрипт, осуществляющая синонимайз.

СТОП-СЛОВА – служебные части речи (частицы, междометия и т.п.), которые не учитываются поисковыми системами при добавлении отсканированного текста статьи в базу поисковой системы.

ТРАСТ – совокупность показателей сайта, определяющих степень доверия поисковой системы этому сайту.

ТИЦ (ТЕМАТИЧЕСКИЙ ИНДЕКС ЦИТИРОВАНИЯ) – показатель авторитетности сайта у поисковой системы Яндекс.

ТРАФИК – количество передаваемых данных, в SEO под трафиком понимается количество посетителей сайта.

ФРИЛАНС – свободная работа, удаленная работа, как правило, осуществляемая через Интернет.

ФРИЛАНСЕР – человек, занимающийся фрилансом.

ФИЛЬТР – совокупность критериев, по которым происходит отбор сайтов.

ХОСТИНГ – дисковое пространство, на котором располагаются файлы веб-узлов.

ХОСТИНГ-ПРОВАЙДЕР – компания, предоставляющая услуги хостинга.

ЧИСЛОВОЙ АДРЕС ДОМЕНА – это уникальный IP-адрес домена, состоящий из чисел, разделенных точками. Например, 111.222.33.44. Если в адресной строке браузера ввести числовой адрес, то откроется ресурс, соответствующий данному адресу.

ЮЗАБИЛИТИ – удобство пользования сайтом или отдельными его составляющими (например, меню и т.п.).

ЭЛЕКТРОННАЯ КОММЕРЦИЯ – продажа товаров через Интернет.

UNIX ХОСТИНГ – это хостинг, базирующийся на платформе Unix.

WINDOWS ХОСТИНГ – хостинг, базирующийся на платформе

Windows.

FTP КЛИЕНТ (МЕНЕДЖЕР) – программа, предназначенная для соединения по протоколу ftp.

PR (PAGE RANK) – показатель авторитетности сайта у поисковой системы Гугл.

DOFOLLOW– тег, открывающий ссылки для индексации.

NOINDEX – тег, запрещающий поисковому роботу индексировать определенную часть сайта. Чтобы установить запрет, необходимо поместить необходимую часть сайта между открывающим и закрывающим тегами noindex таким образом: <noindex>часть сайта</noindex>

Список литературы

1. Исси Коэн Л. Полный справочник по HTML, CSS и JavaScript [Текст] = The Web Programmer's Desk Reference : справочное издание / Л. Исси Коэн, Д. Исси Коэн. – М. : ЭКОМ Паблишерз, 2007. – 2007 с.

2. Лещев Д. Создание интерактивного web-сайта [Текст] : учеб. курс / Д. Лещев. – СПб. : Питер, 2003. – 544 с.

3. Матросов А. В. HTML 4.0 [Текст] / А. В. Матросов, А. О. Сергеев, М. П. Чаунин. – СПб. : БХВ-Санкт-Петербург, 2001. – 671 с.

4. Морис Б. HTML в действии [Текст] : учеб. пособ. / Б. Морис. – СПб. : Питер, 1997. – 252 с.

5. Пайк М. Internet в подлиннике [Текст] : руководство : пер. с англ. / М. Пайк ; науч. ред. А. Н. Тихонов. – СПб. : BNV-Санкт-Петербург, 1996. – 640 с.

6. Прохоренок Н. А. HTML, JavaScript, PHP и MySQL. Джентельменский набор Web-мастера [Текст] : научное издание / Н. А. Прохоренок. – СПб. : БХВ-Петербург, 2008. – 622 с.

7. Румянцев Д. Сам себе Web-программист. Практикум создания качественного Web-сайта [Текст] / Д. Румянцев. – Москва : ИНФРА-М, 2001. – 207 с.

8. Симонович С. Новейший самоучитель по работе в Интернете [Текст] : самоучитель / С. Симонович, Г. Евсеев. – М. : Информком-Пресс : ДЕСС КОМ, 2000. – 528 с.

9. Способы обработки электронных документов в Интернет : визуальный редактор гипертекстовых документов DreamWeaver [Текст] : учеб. пособ. / сост. Д. Э. Ситников [и др.]. – Х. : ХГАК, 2003. – 186 с.

10. Хеслоп Б. HTML с самого начала [Текст] / Б. Хеслоп – СПб. : Питер, 1997. – 406 с.

11. Чебыкин Р. И. Самоучитель HTML и CSS. Современные технологии [Текст] : научное издание / Р. И. Чебыкин. – СПб. : БХВ-Петербург, 2008. – 608 с.

12. Чиртик А. А. HTML. Популярный самоучитель [Текст] / А. А. Чиртик. – СПб. : Питер, 2006. – 224 с.

13. Шапошников И. В. Интернет-программирование [Текст] / И. В. Шапошников. – СПб. : БХВ-Петербург, 2000. – 214 с.

ПРИЛОЖЕНИЯ

Приложение 1. Пример оформления титульного листа

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
"ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

Кафедра інтелектуальних комп'ютерних систем

Лабораторна робота № __
на тему

ТЕМА ЛАБОРАТОРНОЇ РОБОТИ

Виконав (Виконала)
студент (студентка)
групи ІФ-___
Прізвище та ініціали студента

Перевірив (Перевірила)
Посада
Прізвище та ініціали викладача

Харків-20__

Приложение 2. Таблица цветов

Код цвета	Название CSS	Название цвета
#F0F8FF	aliceblue	блекло-голубой
#FAEBD7	antiquewhite	античный белый
#00FFFF	aqua	синий
#7FFF00	chartreuse	фисташковый
#F5F5DC	azure	лазурь
#F5F5DC	beige	бежевый
#FFE4C4	bisque	бисквитный
#FFF8DC	cornsilk	темно-зеленый
#FFEBCD	blanchedalmond	светло-кремовый
#0000FF	blue	голубой
#8A2BE2	blueviolet	светло-фиолетовый
#B8860B	darkgoldenrot	темный красно-золотой
#006400	darkgreen	темно-зеленый
#8B008B	darkmagenta	темный фуксин
#FF8C00	darkorange	темно-оранжевый
#8B0000	darkred	темно-красный
#8FBC8F	darkseagreen	темный морской волны
#2F4F4F	darkslategray	темный сине-серый
#9400D3	darkviolet	темно-фиолетовый
#00BFFF	deepskyblue	темный небесно-голубой
#1E90FF	dodgerblue	тускло-васильковый
#FFFAF0	floralwhite	цветочно-белый
#FF00FF	fuchsia	фуксии

Продолжение Приложения 2

Код цвета	Название CSS	Название цвета
#DCDCDC	gainsboro	гейнсборо
#DAA520	goldenrod	красного золота
#008000	green	зеленый
#F0FFF0	honeydew	свежего меда
#CD5C5C	indianred	ярко-красный
#FFFFF0	ivory	слоновой кости
#E6E6FA	lavender	бледно-лиловый
#7CFC00	lawngreen	зеленой лужайки
#ADD8E6	lightblue	светло-голубой
#E0FFFF	lightcyan	светло-циановый
#90EE90	lightgreen	светло-зеленый
#FFB6C1	lightpink	светло-розовый
#20B2AA	lightseagreen	светлый морской волны
#778899	lightslategray	светлый сине-серый
#FFFFE0	lightyellow	светло-желтый
#32CD32	limegreen	зеленовато-известковый
#FF00FF	фуксин	blanchedalmond
#66CDAA	mediumaquamarine	умеренно-аквамариновый
#3CB371	mediumseagreen	умеренный морской волны
#BA55D3	mediumorchid	умеренно-орхидейный
#00FA9A	mediumspringgreen	умеренный сине-серый
#0C71585	mediumvioletred	умеренный красно-фиолетовый
#0F5FFFA	mintcream	мятно-кремовый

Продолжение Приложения 2

Код цвета	Название CSS	Название цвета
#FFE4B5	moccasin	болотный
#000080	navy	морской
#808000	olive	оливковый
#FFA500	orange	оранжевый
#DA70D6	orchid	орхидейный
#98FB98	palegreen	бледно-зеленый
#DB7093	palevioletred	бледный красно-фиолетовый
#FFDAB9	peachpuff	персиковый
#FFC0CB	pink	розовый
#B0E0E6	powderblue	туманно-голубой
#FF0000	red	красный
#4169E1	royalblue	королевский голубой
#FA8072	salmon	оранжево-розовый
#2E8B57	seagreen	морской зеленый
#A0522D	sienna	охра
#87CEEB	skyblue	небесно-голубой
#708090	slategray	сине-серый
#00FF7F	springgreen	весенне-зеленый
#D2B48C	tan	желто-коричневый
#D8BFD8	thistle	чертополоха
#40E0D0	turquoise	бирюзовый
#F5DEB3	wheat	пшеничный
#F5F5F5	whitesmoke	белый дымчатый

Продолжение Приложения 2

Код цвета	Название CSS	Название цвета
#9ACD32	yellowgreen	желто-зеленый
#A52A2A	brown	коричневый
#DEB887	burlywood	старого дерева
#5F9EA0	cadetblue	блеклый серо-голубой
#FF7F50	coral	коралловый
#D2691E	chocolate	шоколадный
#6495ED	cornflowerblue	васильковый
#000000	black	черный
#DC143C	crimson	малиновый
#00FFFF	cyan	циан
#00008B	darkblue	темно-голубой
#008B8B	darkcyan	темный циан
#A9A9A	darkgray	темно-серый
#BDB76B	darkkhaki	темный хаки
#556B2F	darkolivegreen	темно-оливковый
#9932CC	darkorchid	темно-орхидейный
#E9967A	darksalmon	темно-оранжево-розовый
#483D8B	darkslateblue	темный сине-серый
#00CED1	darkturqueise	темно-бирюзовый
#FF1493	deeppink	темно-розовый
#696969	dimgray	тускло-серый
#B22222	firebrick	огнеупорного кирпича
#228B22	forestgreen	лесной зеленый

Продолжение Приложения 2

Код цвета	Название CSS	Название цвета
#F8F8FF	ghostwhite	туманно-белый
#FFD700	gold	золотой
#808080	gray	серый
#ADFF2F	greenyellow	желто-зеленый
#FF69B4	hotpink	ярко-розовый
#4B0082	indigo	индиго
#F0E68C	khaki	хаки
#FFF0F5	lavenderblush	бледный розово-лиловый
#FFFACD	lemonchiffon	лимонный
#F08080	lightcoral	светло-коралловый
#FAFAD2	lightgoldenrodyellow	светлый золотисто-желтый
#D3D3D3	lightgrey	светло-серый
#FFA07A	lightsalmon	светлый оранжево-розовый
#87CEFA	lightskyblue	светлый небесно-голубой
#B0C4DE	lightsteelblue	светло-стальной
#00FF00	lime	цвета извести
#FAF0E6	linen	льняной
#800000	maroon	оранжево-розовый
#0000CD	mediumblue	умеренно-голубой
#7B68EE	mediumslateblue	умеренный серо-голубой
#9370DB	mediumpurple	умеренно-пурпурный
#48D1CC	mediumturquose	умеренно-бирюзовый
#191970	midnightblue	полуночный синий

Продолжение Приложения 2

Код цвета	Название CSS	Название цвета
#FFE4E1	mistyrose	туманно-розовый
#FFDEAD	navajowhite	грязно-серый
#6B8E23	olivedrab	цвет хаки
#FF4500	orangered	оранжево-красный
#EEE8AA	palegoldenrod	бледно-золотистый
#AFEEEE	paleturquoise	бледно-бирюзовый
#FFEFD5	papayawhip	дынный
#CD853F	peru	коричневый
#DDA0DD	plum	сливовый
#800080	purple	пурпурный
#BC8F8F	rosybrown	розово-коричневый
#F4A460	sandybrown	рыже-коричневый
#FFF5EE	seashell	морской пены
#C0C0C0	silver	серебристый
#6A5ACD	slateblue	серо-голубой
#FFFAFA	snow	снежный
#4682B4	steelblue	сине-стальной
#008080	teal	чайный
#FF6347	tomato	томатный
#EE82EE	violet	фиолетовый
#FFFFFF	white	белый
#FFFF00	yellow	желтый
#7FFFD4	aquamarine	аквамарин

Приложение 3. Таблица специальных символов

Имя	Код	Вид	Описание
"	"	"	двойная кавычка
&	&	&	амперсанд
<	<	<	знак 'меньше'
>	>	>	знак 'больше'
			неразрывный пробел
¡	¡	¡	перевернутый восклицательный знак
¢	¢	¢	цент
£	£	£	фунт стерлингов
¤	¤	¤	денежная единица
¥	¥	¥	иена или юань
¦	¦	¦	разорванная вертикальная черта
§	§	§	параграф
¨	¨	¨	умляут
©	©	©	знак copyright
ª	ª	ª	женский порядковый числитель
«	«	«	левая двойная угловая скобка
¬	¬	¬	знак отрицания
			место возможного переноса
®	®	®	знак зарегистрированной торговой марки
¯	¯	¯	верхняя горизонтальная черта
°	°	°	градус
±	±	±	плюс-минус
²	²	²	"в квадрате"
³	³	³	"в кубе"
´	´	´	знак ударения
µ	µ	µ	микро
¶	¶	¶	символ параграфа
·	·	·	точка

Продолжение Приложения 3

Имя	Код	Вид	Описание
¸	¸	¸	седиль (орфографический знак)
¹	¹	¹	верхний индекс 'один'
º	º	º	мужской порядковый числитель
»	»	»	правая двойная угловая скобка
¼	¼	¼	одна четвертая
½	½	½	одна вторая
¾	¾	¾	три четвертых
¿	¿	¿	перевернутый вопросительный знак
À	À	À	латинская заглавная A с тупым ударением
Á	Á	Á	латинская заглавная A с острым ударением
Â	Â	Â	латинская заглавная A с диакритическим знаком над гласной
Ã	Ã	Ã	латинская заглавная A с тильдой
Ä	Ä	Ä	латинская заглавная A с двумя точками
Å	Å	Å	латинская заглавная A с верхним кружком
Æ	Æ	Æ	латинские заглавные символы AE вместе
Ç	Ç	Ç	латинская заглавная C с седилем
È	È	È	латинская заглавная E с тупым ударением
É	É	É	латинская заглавная E с острым ударением
Ê	Ê	Ê	латинская заглавная E с диакритическим знаком над гласной
Ë	Ë	Ë	латинская заглавная E с двумя точками
Ì	Ì	Ì	латинская заглавная I с тупым ударением
Í	Í	Í	латинская заглавная I с острым ударением
Î	Î	Î	латинская заглавная I с диакритическим знаком над гласной
Ï	Ï	Ï	латинская заглавная I с двумя точками
Ð	Ð	Ð	латинская заглавная D с черточкой
Ñ	Ñ	Ñ	латинская заглавная N с тильдой

Продолжение Приложения 3

Имя	Код	Вид	Описание
Ò	Ò	Ò	латинская заглавная O с тупым ударением
Ó	Ó	Ó	латинская заглавная O с острым ударением
Ô	Ô	Ô	латинская заглавная O с диакритическим знаком над гласной
Õ	Õ	Õ	латинская заглавная O с тильдой
Ö	Ö	Ö	латинская заглавная O с двумя точками
×	×	×	знак умножения
Ø	Ø	Ø	латинская заглавная O со штрихом
Ù	Ù	Ù	латинская заглавная U с тупым ударением
Ú	Ú	Ú	латинская заглавная U с острым ударением
Û	Û	Û	латинская заглавная U с диакритическим знаком
Ü	Ü	Ü	латинская заглавная U с двумя точками
Ý	Ý	Ý	латинская заглавная Y с острым ударением
Þ	Þ	Þ	латинская заглавная THORN
à	à	à	латинская строчная a с тупым ударением
á	&##225;	á	латинская строчная a с острым ударением
â	&##226;	â	латинская строчная a с диакритическим знаком
ã	ã	ã	латинская строчная a с тильдой
ä	ä	ä	латинская строчная a с двумя точками
å	å	å	латинская строчная a с верхним кружком
æ	æ	æ	латинская строчные буквы ae
ç	ç	ç	латинская строчная c с седилем
è	è	è	латинская строчная e с тупым ударением
é	é	é	латинская строчная e с острым ударением
ê	ê	ê	латинская строчная e с диакритическим знаком
ë	ë	ë	латинская строчная e с двумя точками
ì	ì	ì	латинская строчная I с тупым ударением

Продолжение Приложения 3

Имя	Код	Вид	Описание
í	í	í	латинская строчная I с острым ударением
î	î	î	латинская строчная I с диакритическим знаком
ï	ï	ï	латинская строчная I с двумя точками
ð	ð	ð	латинская строчные символы eth
ñ	ñ	ñ	латинская строчная N с тильдой
ò	ò	ò	латинская строчная O с тупым ударением
ó	ó	ó	латинская строчная O с острым ударением
ô	ô	ô	латинская строчная O с диакритическим знаком
õ	õ	õ	латинская строчная I с тильдой
ö	ö	ö	латинская строчная I с двумя точками
÷	÷	÷	знак деления
ø	ø	ø	латинская строчная O со штрихом
ù	ù	ù	латинская строчная U с тупым ударением
ú	ú	ú	латинская строчная U с острым ударением
û	û	û	латинская строчная U с диакритическим знаком
ü	ü	ü	латинская строчная U с двумя точками
ý	ý	ý	латинская строчная Y с острым ударением
þ	þ	þ	латинская строчная thorn
ÿ	ÿ	ÿ	латинская строчная Y с двумя точками
ƒ	ƒ	;	знак функции
Символы для букв			
ˆ	ˆ	;	диакритический знак над гласной
˜	˜	;	тильда
Греческие буквы			
Α	Α	;	заглавная альфа
Β	Β	;	заглавная бета

Продолжение Приложения 3

Имя	Код	Вид	Описание
Γ	Γ	;	заглавная гамма
Δ	Δ	;	заглавная дельта
Ε	Ε	;	заглавная эпсилон
Ζ	Ζ	;	заглавная дзета
Η	Η	;	заглавная эта
Θ	Θ	;	заглавная тета
Ι	Ι	;	заглавная иота
Κ	Κ	;	заглавная каппа
Λ	Λ	;	заглавная лямбда
Μ	Μ	;	заглавная мю
Ν	Ν	;	заглавная ню
Ξ	Ξ	;	заглавная кси
Ο	Ο	;	заглавная омикрон
Π	Π	;	заглавная пи
Ρ	Ρ	;	заглавная ро
Σ	Σ	;	заглавная сигма
Τ	Τ	;	заглавная тау
Υ	Υ	;	заглавная ипсилон
Φ	Φ	;	заглавная фи
Χ	Χ	;	заглавная хи
Ψ	Ψ	;	заглавная пси
Ω	Ω	;	заглавная омега
α	α	;	строчная альфа
β	β	;	строчная бета
γ	γ	;	строчная гамма
δ	δ	;	строчная дельта
ε	ε	;	строчная эпсилон
ζ	ζ	;	строчная дзета

Продолжение Приложения 3

Имя	Код	Вид	Описание
η	η	;	строчная эта
θ	θ	;	строчная тета
ι	ι	;	строчная иота
κ	κ	;	строчная каппа
λ	λ	;	строчная лямбда
μ	μ	;	строчная мю
ν	ν	;	строчная ню
ξ	ξ	;	строчная кси
ο	ο	;	строчная омикрон
π	π	;	строчная пи
ρ	ρ	;	строчная ро
ς	ς	;	строчная сигма (final)
σ	σ	;	строчная сигма
τ	τ	;	строчная тау
υ	υ	;	строчная ипсилон
φ	φ	;	строчная фи
χ	χ	;	строчная хи
ψ	ψ	;	строчная пси
ω	ω	;	строчная омега
Общая пунктуация			
–	–	–	тире
—	—	—	длинное тире
‘	‘	‘	левая одиночная кавычка
’	’	’	правая одиночная кавычка
‚	‚	,	нижняя одиночная кавычка
“	“	“	левая двойная кавычка
”	”	”	правая двойная кавычка
„	„	„	нижняя двойная кавычка

Продолжение Приложения 3

Имя	Код	Вид	Описание
Прочие символы			
†	&#8224	†	латинский крест
‡	&#8225	‡	двойной крест
•	•	•	маленький черный кружок
…	…	…	многоточие
	‰	‰	знак промилле
′	′	;	одиночный штрих – минуты
″	″	;	двойной штрих – секунды
‾	‾	;	надчеркивание
⁄	⁄	;	косая дробная черта
€	€	?	евро
	№	№	знак номера
™	™	™	знак торговой марки
◊	&#9674	;	ромб
	&#9675	;	круг
Стрелки			
←	←	;	стрелка влево
↑	↑	;	стрелка вверх
→	→	;	стрелка вправо
↓	↓	;	стрелка вниз
↔	↔	;	стрелка влево-вправо
	↕	;	стрелка вверх-вниз
Масти			
♠	♠	;	знак масти 'пики'
♣	♣	;	знак масти 'трефы'
♥	♥	;	знак масти 'червы'
♦	♦	;	знак масти 'бубны'

Навчальне видання

БОРИСОВА Наталя Володимирівна
КАНІЩЕВА Ольга Валеріївна

ОСНОВИ ВЕБ-ТЕХНОЛОГІЙ

Навчальний посібник
для студентів спеціальності "Прикладна лінгвістика"

Російською мовою

Відповідальний за випуск проф. Н. В. Шаронова
Роботу до видання рекомендував проф. О. В. Горілий

Редактор Н. В. Верстюк

План 2013 р., поз. 63

Підп. до друку __.__.__. Формат 60×84 1/16. Папір друк. №2.
Друк – ризографія. Гарнітура Times New Roman. Ум. друк. арк. ___.
Наклад ____ прим. Зам № _____. Ціна договірна.

Видавничий центр НТУ "ХПІ".
Свідоцтво про державну реєстрацію ДК № 3657 від 24.12.2009 р.
61002, Харків, вул. Фрунзе, 21

Друкарня НТУ "ХПІ".
61002, Харків, вул. Фрунзе, 21