Е.В. Киргизова, А.В. Рубцов

WEB-ТЕХНОЛОГИИ: от теории к практике

Министерство науки и высшего образования РФ
Сибирский федеральный университет
Лесосибирский педагогический институт

Е.В. Киргизова, А.В. Рубцов

WEB-ТЕХНОЛОГИИ: от теории к практике

Учебное пособие

Рекомендовано УМО РАЕ по классическому университетскому и техническому образованию в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению 44.03.01 – «Педагогическое образование» профиль «Информатика»; по направлению 44.03.05 – «Педагогическое образование» профиль «Информатика и экономика; по направлению 09.03.02 – «Информационные системы и технологии» профиль обучения «Информационно-управляющие системы» (Протокол № 672 от 27.11.2017 г.)

Красноярск-Лесосибирск, 2018

УДК 004.43 (075.8)
ББК 32.973.26 – 018.1я73
К 43

Рецензенты:
Н.И. Пак, д-р пед. наук, профессор (Красноярский государственный педагогический институт им. В.П. Астафьева);
К.В. Сафонов, д-р. физ.-мат. наук, профессор**,** академик Международной академии информатизации (Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева)

К43 Киргизова Е.В. Web-технологии: от теории к практике: учеб. пособие / Е.В. Киргизова, А. В. Рубцов. – Красноярск: Сибирский федеральный университет, 2018. – 160 с.

Учебное пособие содержит необходимые теоретические сведения о принципах размещения и передачи информации в Интернете, концепциях и возможностях наиболее популярных технологий современной Web-разработки, а также базовых элементах обеспечения информационной безопасности Web-приложений.

Пособие предназначено для изучения Web-технологий студентами вузов направления по направлению 44.03.01 - «Педагогическое образование» профиль «Информатика»; по направлению 44.03.05 - «Педагогическое образование» профиль «Информатика и экономика; по направлению 09.03.02 – «Информационные системы и технологии» профиль «Информационно-управляющие системы».

ISBN 978-5-7638-3808-4

УДК 004.43 (075.8)
ББК 32.973.26 – 018.1я73

© Лесосибирский педагогический институт – филиал Сибирского федерального университета, 2018

ВВЕДЕНИЕ

Современный Интернет – сложная и высокотехнологичная система, позволяющая пользователю общаться с людьми, находящимися в любой точке земного шара, быстро и комфортно отыскивать любую необходимую информацию, публиковать для всеобщего сведения данные, которые он хотел бы сообщить всему миру.

Web-технологии стремительно развиваются, проникая в самые разнообразные сферы профессиональной деятельности, в том числе и экономической. Для компаний присутствие в Интернете – это возможность рассказать о своих товарах и услугах, найти потенциальных партнеров и клиентов, а также снизить издержки за счет интернет-торговли, использования «облачных» сервисов. Даже такие традиционно замкнутые системы, как промышленные автоматизированные системы управления производством, в том числе и в критических отраслях, также в большинстве случаев прямо или косвенно подключены к Интернету.

Актуальность учебного пособия обуславливается бурным развитием инструментов и средств разработки Web-сайтов и Web-приложений и необходимостью выполнения студентами большого объёма самостоятельной работы для эффективного освоения всех этих технологий.

Основная цель пособия – помочь студенту быстро и эффективно освоить:

- теоретические основы функционирования глобальной компьютерной сети Интернет;
- историю и перспективы его развития;
- инструменты, технологии и методы разработки Web-сайтов и Web-приложений.

Данное учебное пособие посвящено рассмотрению набора популярных в настоящее время средств поддержки и разработки распределенных Web-приложений, в который входят языки HTML 4.0, JavaSript, PHP 5, SQL, каскадные таблицы стилей CSS 2.0, объектная модель браузера DOM (Document Object Model).

Раздел 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Глава 1. Введение в технологию создания Web-сайтов

1.1. Понятие Web-сайта

Информация, доступная пользователям Интернета, располагается на компьютерах (Web-серверах), на которых установлено специальное программное обеспечение. Значительная часть этой информации организованна в виде Web-сайтов. Каждый из них имеет свое имя (адрес) в Internet. Web-сайт – это информация, представленная в определенном виде, которая располагается на Web-сервере и имеет свое имя (адрес). Для просмотра Web-сайтов на компьютере пользователя применяются специальные программы, которые называются браузерами.

Наиболее распространенными браузерами в настоящее время являются Internet Explorer и Netscape Navigator. В зависимости от того, какое имя (адрес) сайта мы зададим в строке «**Адрес»**, браузер будет загружать в свое окно соответствующую информацию.

Web-сайт состоит из связанных между собой Web-страниц. Web-страница представляет собой текстовый файл с расширением *.htm, который содержит текстовую информацию и специальные команды – HTML-коды, определяющие, в каком виде эта информация будет отображаться в окне браузера. Вся графическая, аудио- и видеоинформация непосредственно в Web-страницу не входит и представляет собой отдельные файлы с расширениями *.gif, *.jpg (графика), *.mid, *.mp3 (звук), *.avi (видео). В HTML-коде страницы содержатся только указания на такие файлы (рис. 1).

Рис. 1. Структура Web-сайта

Каждая страница Web-сайта также имеет свой интернет адрес, который состоит из адреса сайта и имени файла, соответствующего данной странице.

Таким образом, **Web-сайт –** это информационный ресурс, состоящий из связанных между собой гипертекстовых документов (Web-страниц), размещенный на Web-сервере и имеющий индивидуальный адрес.

1.2. Классификация Web-сайтов

В настоящее время во всемирной паутине размещено несколько миллионов Web-сайтов и их число постоянно растет.

Таблица 1. Классификация сайтов

Классификационный признак	Типы сайтов
По доступности сервисов	– *Открытые* – все сервисы полностью доступны для любых посетителей. – *Полуоткрытые* – для доступа необходимо зарегистрироваться (обычно бесплатно). – *Закрытые* – полностью закрытые служебные сайты организаций (в том числе корпоративные сайты), личные сайты частных лиц. Такие сайты доступны для узкого круга людей. Доступ новым людям даётся через инвайты (приглашения).
По природе содержимого	– *Статические* – всё содержимое заранее подготавливается. Пользователю выдаются файлы в том виде, в котором они хранятся на сервере. – *Динамические* – содержимое генерируется специальными скриптами (программами) на основе других данных из любого источника.
По физическому расположению	– *Внешние сайты* сети Интернет. – *Локальные сайты* – доступны только в пределах локальной сети. Это могут как корпоративные сайты организаций, так как и сайты частных лиц в локальной сети провайдера.
По схеме представления информации, её объёму и категории решаемых задач	- *Интернет-представительства владельцев* (торговля и услуги, не связанные напрямую с Интернетом): – Сайт-визитка – содержит самые общие данные о владельце сайта (организация или индивидуальный предприниматель). Вид деятельности, история, прайс-лист, контактные данные, реквизиты, схема проезда. Специалисты размещают своё резюме. То есть подробная визитная карточка. – Каталог продукции – в каталоге присутствует подробное описание товаров/услуг, сертификаты, технические и потребительские данные, отзывы экспертов и т. д. На таких сайтах размещается информация о товарах/услугах, которую невозможно поместить в прайс-лист. – Интернет-магазин – Web-сайт с каталогом продукции, с помощью которого клиент может заказать нужные ему товары. Используются различные системы расчётов: от пересылки товаров наложенным платежом или автоматической пересылки счета по факсу до расчётов с помощью пластиковых карт. – Промо-сайт – сайт о конкретной торговой марке или продукте, на таких сайтах размещается исчерпывающая информация о бренде, различных рекламных акциях (конкурсы, викторины, игры и т. п.). – Сайт-квест – Iнтернет-ресурс, на котором организовано соревнование по разгадыванию последовательности взаимосвязанных логических загадок. – *Информационные ресурсы*: – Тематический сайт – Web-сайт, предоставляющий исчерпывающую информацию о какой-либо теме. – Тематический портал – это очень большой web-ресурс, который предоставляет исчерпывающую информацию по определённой тематике. Порталы похожи на тематические сайты, но дополнительно содержат средства взаимодействия с пользователями и позволяют пользователям общаться в рамках портала (форумы, чаты) - это среда существования пользователя. – *Веб-сервис* – обычно решает конкретную пользовательскую задачу напрямую связанную с сетью Интернет: – Поисковые сервисы (например, Яндекс, Google). – Почтовый сервис. – Веб-форумы. – Блоговый сервис. – Фото хостинг (например, Flickr, ImageShack, Panoramio, Photobucket). – Хранение видео (например, YouTube, RuTube). – Доска объявлений. – Каталог сайтов (например, OpenDirectoryProject).
По отношению к посетителю	– Вовлекающий сайт. – Безразличный к посетителю.

1.3/ Этапы разработки Web-сайта

Главными задачами при разработке любого Web-сайта являются четкая организация структуры сайта и определение его информационного наполнения. Другими словами, на первом этапе необходимо создать информационную модель Web-сайта.

Выделяют следующие этапы разработки Web-сайта: планирование, реализация, тестирование, публикация, рекламирование, сопровождение.

Планирование является первым и, вероятно, наиболее важным этапом создания хорошего Web-сайта. На стадии планирования определяется следующее:

– цель создания Web-сайта;

– характер содержимого;

– структура (jт того, как организована подача материала на Web-сайте, зависит «путь», который должен пройти пользователь в поисках нужной ему информации);

– особенности оформления.

Любую страницу можно оценить по трем параметрам: *контенту*, *внешнему виду и навигации*. Одно должно дополнять другое и ни в коем случае не подавлять. Если Вы публикуете свой труд, рассчитываемый на долгое вдумчивое чтение, тогда позаботьтесь о читабельности, постарайтесь исключить отвлекающие динамические эффекты, подберите правильное, не утомляющее цветовое сочетание фона и текста, постарайтесь задать стиль, соответстветствующие содержанию. И наоборот: сократите текстовые блоки до минимума, если Вы создаете сайт, дающий посетителю прежде всего визуальную и другую мультимедийную информацию, подключайте всю Вашу фантазию и доступные средства реализации.

Реализация – это и есть работа по созданию сайта. На этом этапе проводится подготовка текстового и графического материала (печать, сканирование). Материал разбивается по файлам в соответствии со структурой. Организуются ссылки между файлами сайта. Рекомендуется создать **шаблон-заготовку** страницы с основными структурными областями и стилевым оформлением и использовать ее для создания всех страниц узла. Меняйте в каждой новой странице только содержимое и адресацию ссылок, такая организация работы сократит время, потраченное на каждую из них. Помните, что посетитель может попасть прямо из поисковой системы на любую из ваших страниц - и в этом случае важно показать ему, что она является частью целого сайта, дать ему возможность перейти по ссылке на главную страницу и просмотреть остальные разделы.

При создании Web-страниц необходимо учитывать, что разработанный Вами Web-сайт может выглядеть на компьютерах разных пользователей по-разному. Это зависит от многих параметров – типа браузера клиента, установок операционной системы, аппаратных ресурсов компьютера и т.п.

Тестирование

Завершив работу по размещению страниц на Web-сайте, необходимо выполнить тестирование. Оно состоит из двух этапов: тестирование на работоспособность и тестирование на удобство пользования интерфейсом.

На этапе тестирования на работоспособность проверяют, как

функционирует Web-сайт, используя те же условия, при которых с ним будет работать пользователь. Поработайте с Web-сайтом в различных браузерах и посмотрите, как выглядит Ваш сайт в каждом из них. Постарайтесь оценить время загрузки страниц, что очень важно. Для тестирования на удобство пользования интерфейсом крупные компании приглашают специальные группы людей. Вам можно пригласить своих друзей и, не давая им никаких инструкций, посмотреть, как они будут пользоваться вашим Web- сайтом.

Обратите внимание на то, как они перемещаются по Web-сайту. Где возникают паузы? Когда пользователи испытывают трудности? И при этом не подсказывайте им, не давайте никаких указаний! Такие наблюдения дадут вам много ценной информации. А если пользователи будут выполнять неправильные действия, то это уже недостаток вашей разработки и, значит, над Web-сайтом следует еще поработать.

Выслушайте пользователей, может, они подскажут вам некоторые решения возникших проблем.

Публикация

Готовый Web-сайт необходимо опубликовать на Web-сервере, чтобы он был доступен через Интернет. Если ваш сайт создан посредством редактора FrontPage, то на сервере должны быть инсталлированы серверные расширения FrontPage, что обеспечит полную поддержку доступных в FrontPage компонентов, которые были помещены на странице в процессе создания сайта.

Если у вас нет собственного сервера, то в Сети можно найти огромное количество ссылок на **free web pages**, где некоторые провайдеры предоставляют своим клиентам бесплатное место под страницу.

Рекламирование сайта

Существует множество приемов рекламирования сайта: размещение информации о нем на поисковом Web-сайте, организация взаимных ссылок с другими сайтами и т.д.

Как привлечь пользователя? Красиво оформленные страницы Web-сайта – это только половина дела. В первую очередь страницы должны быть содержательными.

Основное требование к содержимому Web-сайта – полнота и достоверность. Информация должна быть представлена таким образом, чтобы пользователь, однажды посетивший Web-сайт, еще не раз обратился к нему.

Акцентируйте внимание пользователя на своей личности или организации, в которой вы работаете, например, создав отдельную страницу, посвященную этой теме. Разместите на домашней странице свою фотографию или логотип организации.

Сопровождение сайта

Содержимое Web-сайта может подвергаться неоднократным изменениям. Важно, чтобы предоставляемая на Web-сайте информация всегда была актуальной, поэтому как можно чаще обновляйте информацию на своем Web-сайте, по возможности расширяйте материал, улучшайте дизайн.

Обязательное правило. Web-сайт должен обновляться не реже одного раза в месяц. В противном случае вы потеряете не только потенциальных, но и уже состоявшихся посетителей.

Рекомендуется создать на своем компьютере копию Web-сайта, вносить в нее изменения и новую версию передавать для размещения на сервере в завершенном виде.

Навигационная схема Web-сайта

Web-сайт состоит из связанных между собой гипертекстовых документов. Гипертекст – это способ хранения и манипулирования информацией, позволяющий устанавливать связи между любыми «информационными единицами». Связь между информационными единицами осуществляется по гиперссылкам. Гиперссылка – это выделенный фрагмент текста, с помощью которого осуществляется переход от одного документа к другому. Обычно гиперссылки выделяют синим цветом и подчеркиванием.

Навигационная схема Web-сайта зависит от его структуры и определяет то, как пользователь будет по нему перемещаться и получать доступ к информации, которую Вы представляете.

Существует несколько видов структурирования информационного материала на Web-сайте:

Линейная структура

Материал весь располагается последовательно (рис. 2).

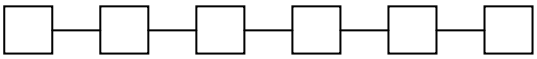


Рис. 2. Линейная структура

Иерархическая структура

Чаще всего структура Web-сайта представляет собой иерархию. При этом сначала создают категории высшего уровня, а затем материал в логическом порядке размещают в категории, которые находятся ниже. Иерархические структуры бывают двух видов: узкая глубокая и широкая неглубокая.

Узкая глубокая иерархия (рис. 3) характеризуется тем, что на верхнем уровне она имеет мало категорий. Для получения нужной информации пользователь вынужден переходить на несколько уровней вниз.

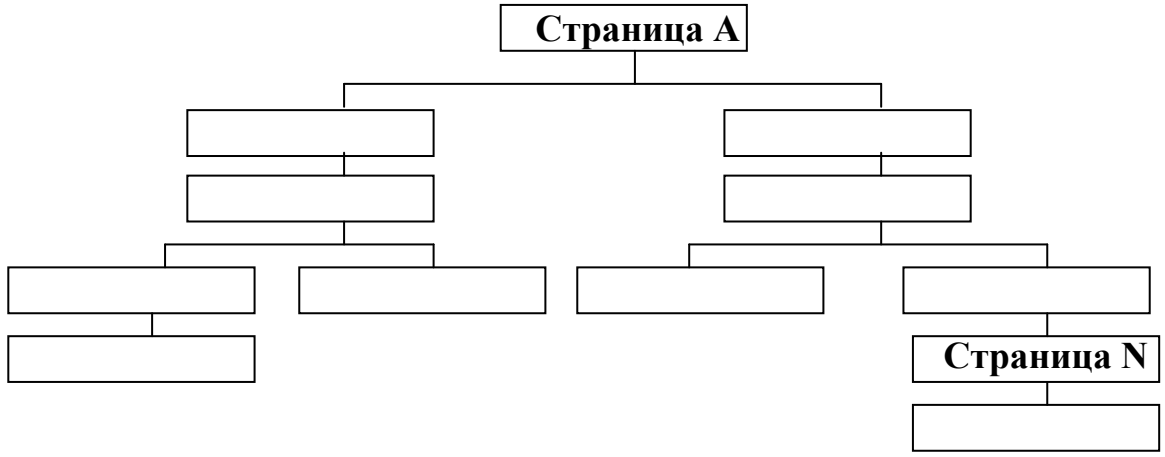


Рис. 3. Узкая глубокая ерархия

Широкая неглубокая иерархия представлена на рис. 4.

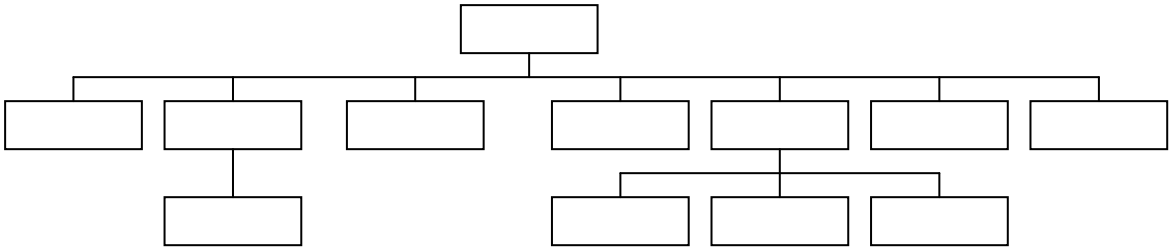


Рис. 4. Широкая неглубокая иерархия

Ни первый, ни второй способ организации информации не является оптимальным. Оптимальной считается такая иерархическая структура, которая состоит из 3-4 уровней.

Нелинейная структура изображена на рис. 5

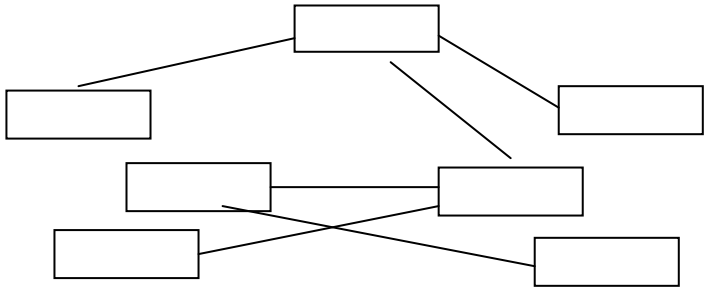


Рис. 5. Нелинейная структура

Смешанная структура

Существуют ситуации, когда представить информацию одним из описанных выше методов не представляется возможным. В этом случае применяют несколько схем одновременно. Однако такой подход имеет недостаток – он требует от пользователя концентрации внимания и дополнительных усилий. Чтобы помочь пользователю составляют карту Web-сайта (site-map). На данной карте схематически изображают структуру размещения информации на Web-сайте.

1.4. История развития web-технологий

Самой простой формулировкой концепции Web 1.0 следует считать «тот Web, который был до Web 2.0».

Переход от Web 1.0 к Web 2.0 является прямым результатом изменений в поведении тех, кто использует Всемирную паутину. Основные тенденции Web 1.0 включали заботы о проблемах безопасности и приватности в одностороннем потоке информации, через Web-сайты, содержащие материал «только для чтения». Характерным для Web 1.0 также являлись компьютерная неграмотность широких масс и распространенность медленных типов подключения к Интернету, вдобавок к ограничениям самого Интернета.

Типичные принципы Web 1.0:

– статичные страницы вместо генерируемого пользователями динамического контента;

– бедная гипертекстовая разметка;

– использование фреймов;

– использование специфичных тегов *HTML*;

– гостевые книги, форумы или чаты;

– указание конкретного разрешения монитора, при котором дизайн сайта отображается корректно;

– крайне редкое и непопулярное использование стилей *CSS* при оформлении страниц сайта.

Появление термина Web 2.0 принято связывать со статьей «Tim O'Reilly - What Is Web 2.0» от 30 сентября 2005 года. В этой статье Тим О'Рейли увязал появление большого числа сайтов, объединенных некоторыми общими принципами, с общей тенденцией развития Интернет – сообщества, и назвал это явление Web 2.0, в противовес устаревшему Web 1.0.

Тим Бернерс-Ли, возглавляющий с 2006 г. крупнейший мировой исследовательский проект по изучению всемирной паутины, назвал термин *Web 2.0* простым жаргоном:

«Никто не знает, что это означает. Если Web 2.0 – это ваши блоги и вики, тогда это значит «пользователи для пользователей». Но это то же самое, что сказать – Web существует, чтобы все люди были вместе».

Понятие Web 2.0 также отразилось и в дизайне. Предпочтительными стали округлость, имитация выпуклых поверхностей, имитация отражений на манер глянцевого пластика современных hi-end – устройств (к примеру, плееры). В целом, восприятие внешнего вида на глаз кажется более приятным. Графика таких сайтов занимает больший объем, нежели при использовании аскетичного дизайна. Отчасти эта тенденция связана с совпавшим по времени выходом новых версий операционных систем, использующих вышеупомянутые идеи.

Наиболее распространенной версией трактовки термина *Веб 3.0* является идентификация его как семантической паутины (Semantic Web). Главная мысль этой концепции базируется на внедрении мета–языка, описывающего содержание сайтов для организации автоматического обмена между серверами.

Семантическая паутина (Semantic Web) – часть глобальной концепции развития сети Интернет, целью которой является реализация возможности машинной обработки информации, доступной во Всемирной паутине. Основной акцент концепции делается на работе с метаданными, однозначно характеризующими свойства и содержание ресурсов Всемирной паутины, вместо используемого в настоящее время текстового анализа документов. Термин впервые введен Тимом Бернерсом-Ли в мае 2001 г. в журнале «Scientific American» и называется им «следующим шагом в развитии Всемирной паутины». В семантической паутине предполагается повсеместное использование, во-первых, универсальных идентификаторов ресурсов (URI), а во-вторых – онтологий и языков описания метаданных.

Эта концепция была принята и продвигается Консорциумом W3C. Для ее внедрения планируется создание сети документов, содержащих метаданные о ресурсах Всемирной паутины и существующие параллельно с ними. Тогда как сами ресурсы предназначены для восприятия человеком, метаданные используются машинами (*поисковыми роботами* и другими интеллектуальными агентами) для проведения однозначных логических заключений о свойствах этих ресурсов.

10 февраля 2004 г. на сайте W3C появляется описание языка «OWL» (язык описания онтологий).

Через полгода новый язык описания онтологий OWL стал поддерживать редактор онтологий Protege – разработка Стэнфордского университета. В это же время Semantic Web начало активно интересоваться международное научное сообщество. В разных изданиях появляется множество статей по Semantic Web.

В 2005 г. на сайте W3 появляется описание RDF/A - синтаксиса, который уже сейчас позволяет встраивать метаданные RDF в документы XHTML.

10 марта 2006 г. выходит RDF/A Primer. Таким образом, Semantic Web был «привязан» к XHTML.

В 2006 г. также завершилась разработка языка запросов к RDF документам с SQL-подобным синтаксисом, его окончательное название - *SPARQL*.

Некоторые авторы дают другую трактовку термину Web 3.0. Так Джейсон Калаканис определяет Web 3.0. как высококачественный контент и сервисы, которые создаются талантливыми профессионалами на технологической платформе Web 2.0.

По сути, Web 3.0 использует технологическую базу Web 2.0:

– AJAX - загрузка данных без перезагрузки самой веб-страницы;

– RIA (Adobe Flex, JavaFX, Microsoft Silverlight);

– XML (eXtensible Markup Language) - язык разметки данных, представляющий собой свод общих синтаксических правил;

– RSS (Really Simple Syndication) - семейство XML – форматов, предназначенных для описания лент новостей, анонсов статей, изменений в блогах и т. п.;

– теги - отображение тегов в виде облака, что значительно упрощает определение пользователем наиболее актуальной информации;

– блоговая структура информации - ленточная подача информации, где поток идет по убыванию сверху-вниз, а метод сортировки задает пользователь.

Главная идея Web 3.0, по мнению этих авторов, состоит в том, чтобы пользователь, который до этого единолично был вовлечен в процесс формирования контента, отныне творит коллективно и его партнерами, помимо других пользователей, являлись эксперты различных направлений, причем статус пользователя может быть изменен на экспертный, равно, как и форма сотрудничества создателя контента и портала. Эксперт должен выступить своеобразным модератором публикуемого контента. По сути, не исключается и возможность платной основы для сотрудничества, но гораздо более важным моментом является появление в порталах формата Web 3.0 «коллективного разума» (wisdom of the *crowds*), вместо господствующего сегодня «группового сумасшествия» (madness of the mobs). Web 3.0 предполагает появление узкоспециализированных ресурсов, где будет произведена агрегация всех необходимых пользователю сервисов и инструментов профессиональной социальной составляющей и будет осуществляться публикация экспертно-модерируемого контента.

Обобщив все вышесказанное, можно выделить общие признаки Веб-концепций.

Web 1.0 – Интернет как информационный портал:

– эксклюзивность информации, необходимо быть первым собственником контента;

– разделение World Wide Web на пригодные для использования каталоги;

– каждый человек имеет свой собственный личный уголок в киберпространстве;

– недостатки: контекст; взаимодействие; масштабируемость;

– Примеры сайтов: Ofoto; Hotmail; Dmoz; GeoCities.

Web 2.0 – Сеть как платформа:

– фокус на сообществах для создания и проверки контента;

– свободная форма организации и классификации контента посредством тэгов;

– создание «интерфейсов» для будущей интеграции (RSS, API);

– недостатки: Персонализация; Мобильность и портативность; Совместимость;

– Примеры сайтов: YouTube; Flickr; Facebook.

Web 3.0 – *Интернет как экспертная система:*

– извлечение проверенной информации;

– мобильность и портативность;

– повсеместное использование RSS и API;

– «дайте мне то, что, как вы думаете, я хочу на основе того, где я был и что делал»;

– примеры сайтов:

Google : универсальный поиск + история пользовательского поиска;

FOAF (friend of a friend) : моя жизнь в RDF;

Wink : социальная поисковая система;

Twitter : «король» микроблогов;
Surface : технология multi-touch для *сенсорных экранов*;
OpenID : единая авторизация в *Интернете*.
Различия концепций представлены в табл. 2:

Таблица 2. Различия концепций Web-технологий

Концепция Свойство	***Web 1.0***	***Web 2.0***	***Web 3.0***
Концепция	Веб только для чтения («the mostly read only web»)	Веб для бурного чтения-записи («the wildly read-write web»)	Портативный индивидуальный Веб («the portable personal web»)
Количество пользователей	45 миллионов глобальных пользователей (1996)	Больше 1 миллиарда глобальных пользователей (2006)	Еще больше
Ориентация	Ориентация на компании (focused on companies)	Ориентация на сообщества (focused on communities)	Ориентация на индивидуальностях (focused on the individual)
Структура данных	Домашние страницы (home pages)	Блоги (blogs)	Lifestreaming-функции (lifestream)
Концепция данных	Владение контентом (owning content)	Обмен контентом (sharing content)	Объединение динамического контента (*consolidating dynamic content*)
Управление знаниями	Britannica Online	Wikipedia	*Интернет*
Технологии	*HTML*, порталы	*XML*, RSS	Технологии «drag and drop» и mashups
Представление	Web формы	Веб приложения	Виджеты (widgets) и гаджеты (*gadgets*)
Классификация	Директории (иерархическое строение) (directories (*taxonomy*))	Тэги (практика совместной категоризации информации (ссылок, фото, видео клипов и т. п.)) (tagging («folksonomy»))	Поведение пользователей (большая сосредоточенность на предпочтениях отдельных лиц) (user behavior («me-onomy»))
Поиск	Netscape	Google	iGoogle, NetVibes
Стоимость рекламы	Просмотр страниц (pages views)	Цена за клик (cost per click)	Активность пользователей (user engagement)
Продвижение	Реклама (advertising)	«Из уст в уста» (word of mouth)	Advertainment

1.5. Web-программирование

Web-*программирование* (Web-разработка) - это бурно развивающийся раздел программирования, ориентированный на разработку динамических Интернет-приложений.

Языки Web-*программирования* делятся на две группы: клиентские и серверные.

Клиентские языки обрабатываются на стороне пользователя (в основном в браузере). Соответственно, обработка скрипта зависит от браузера пользователя, и пользователь имеет полномочия настроить свой браузер так, чтобы тот вообще игнорировал скрипты. При этом если браузер старый, он может не поддерживать тот или иной язык или версию языка, на которую опирался разработчик. С современными браузерами таких проблем возникать не должно, к тому же языки программирования не так уж часто кардинально обновляются (раз в несколько лет) и лучшие из них давно

известны. Код клиентского скрипта может посмотреть каждый, выбрав в меню своего браузера «Исходный код страницы».

Преимущество клиентского языка заключается в том, что обработка скриптов на таком языке может выполняться без отправки документа на сервер. Программа сразу проверит правильное заполнение формы перед отправкой, и, если необходимо, выведет ошибку. Отсюда же вытекает и то ограничение, что с помощью клиентского языка программирования ничто не может быть записано на сервер.

Самым распространенным из клиентских языков признан *JavaScript*, разработчиками которого является компания Netscape совместно с компанией Sun Microsystems. Еще один популярный язык - это VBScript. Помимо этого в последнее время набрали популярность такие технологии, как *AJAX*, *Adobe Flash*, *Microsoft Silverlight* и др.

Серверные языки программирования открывают перед программистом большие просторы для деятельности.

Когда пользователь делает запрос на какую-либо страницу (переходит на нее по ссылке, или вводит адрес в адресной строке своего браузера), то вызванная страница сначала обрабатывается на сервере (то есть выполняются все программы, связанные со страницей) и только потом возвращается к посетителю по сети в виде файла. Этот файл может иметь расширения: *HTML*, *PHP*, *ASP*, *Perl*, *SSI*, *XML*, DHTML, XHTML.

Глава 2. Основы языка HTML/XHTML

2.1. История развития языка HTML

Правила отображения браузерами

Web-страницы - это текстовые файлы, содержащие собственно текст содержимого страницы и команды форматирования, называемые *тегами* (*tag*) или *дескрипторами разметки* языка HTML. Язык разметки гипер-текста HTML (*HyperText Markup Language*) является базовой технологией разработки Web-страниц, которые также называют HTML-документами.

Код HTML интерпретируется браузером (средством просмотра web-страниц), который выводит отформатированное содержимое Web-страницу на экран. Просмотр кода загруженной страницы доступен во всех браузерах, например, в MS Internet Explorer это можно сделать с помощью команды *Вид*/*Просмотр HTML-кода*.

Правила синтаксиса и версии языка HTML

При написании кода страницы названия (имена, идентификаторы) тегов заключаются в угловые скобки < >.

Например, для того, чтобы выделить текст в отдельный абзац, его можно заключить в тег p (от paragraph - абзац):

```
<p>Текст абзаца</p>
```

Большинство тегов используются попарно и называются *парными,* или *тегами-контейнерами*. Первый тег называется *открывающим*, а соответствующий парный ему - *закрывающим*. Закрывающий тег имеет то же имя, что и открывающий, но предваряется символом слеша (/).

Действие тегов-контейнеров распространяется на все содержимое, заключенное между открывающей и закрывающей частями тега. Таким образом, открывающая часть тега отмечает начало действия тега, а закрывающая - конец (прекращение) действия тега.

Теги-контейнеры могут содержать не только текст, но и другие теги, которые в этом случае называют *вложенными*. Число вложенных тегов и уровней вложенности не ограничено.

Например, часть текста абзаца можно выделить жирным шрифтом с помощью тега b (от bold - жирный):

```
<p>Текст абзаца <b>с жирным написанием</b></p>
```

Для тега можно задавать *параметры* (атрибуты, attribute), которые уточняют действие тега, то есть определяют дополнительные характеристики и режимы его функционирования. Наборы допустимых параметров индивидуальны для каждого тега. Параметры указываются обязательно после имени тега и отделяются друг от друга пробелами. Порядок следования параметров в теге произволен.

Большинство параметров требует указания принимаемых значений. Если требуется указать значение, оно записывается после имени параметра через знак равенства = и заключается в двойные кавычки.

Например, текст абзаца можно отцентрировать, задав параметр align со значением center:

Текст абзаца

Если параметр не указан в теге в явном виде, то принимается его значение по умолчанию. Например, если не указывать параметр align для тега p, то по умолчанию текст абзаца будет выровнен по левому краю страницы.

В HTML некоторые параметры не требуют указания значений. Обычно это параметры, которые выключают некоторое свойство.

Например, отключение переноса текста в ячейке таблицы:

<td nowrap>Текст ячейки</td>

Принципиальным отличием закрывающей части тега является то, что она никогда не содержит параметров. Таким образом, она прекращает действие тега вместе со всеми его настройками.

Кроме тегов-контейнеров в HTML, определены теги, выполняющее некоторое одиночное действие, например, вставку рисунка img, горизонтальной линии hr, перевод текста на новую строку br, и поэтому не требуют закрывающей части. Эти теги называются *непарными*.

Исключение из правил записи тегов составляет тег комментария с ограничителями <!-- и -->. Все, что находится внутри этого тега, считается комментарием и не отображается браузером на экране.

Все теги и их атрибуты специфицированы в HTML, их число достаточно ограниченно. С течением времени набор тегов и рекомендации по их использованию менялись. В настоящее время часть тегов и атрибутов, входящих в актуальную спецификацию HTML, помечены как устаревшие и не рекомендованные к использованию.

Официальной спецификации HTML 1.0 не существует. До 1995 года существовало множество неофициальных вариантов HTML. Чтобы стандартная версия отличалась от них, ей сразу присвоили второй номер.

Версия 3 была предложена Консорциумом Всемирной паутины (World Wide Web Consortium, W3C) в марте 1995 г. и обеспечивала много новых возможностей, таких как создание таблиц, «обтекание» изображений текстом и отображение сложных математических формул. Даже при том, что этот стандарт был совместим со второй версией, реализация его была сложна для браузеров того времени. Версия 3.1 официально никогда не предлагалась, и следующей версией стандарта HTML стала 3.2, в которой были опущены многие нововведения версии 3.0, но добавлены нестандартные элементы, поддерживаемые популярными в то время браузерами Netscape и Mosaic.

Версии языка HTML:

– HTML 2.0, одобренный как стандарт 22 сентября 1995 г.;

– HTML 3.2- 14 января 1997 г.;

– HTML 4.0 - 18 декабря 1997 г.;

– HTML 4.01 - 24 декабря 1999 г.;

– HTML 5- 28 октября 2014 г.;

– HTML 5.1 начал разрабатываться примерно 19 декабря 2012 г.

HTML версии 4.0 содержит много элементов, специфичных для отдельных браузеров, но в то же время произошла некоторая «очистка» стандарта. Многие элементы были отмечены как устаревшие и нерекомендованные (deprecated). В частности, были помечены как устаревшие такие теги, как font, используемый для изменения свойств

шрифта, u, задающий подчеркивание символов текста (вместо них рекомендуется использовать таблицы стилей каскадные таблицы стилей CSS). С другой стороны, HTML 4.0 стал поддерживать большее количество опций мультимедиа, языков скриптов, каскадных таблиц стилей.

В версии HTML 4.01 многие отмеченные как устаревшие теги (определение шрифта и базового шрифта font, basefont, центрированный абзац center, подчеркнутый u и зачеркнутый текст s, strike и др.), а также ряд атрибутов оформления (таких как выравнивание align, задание фоновых цветов bgcolor, упомянутый ранее nowrap и др.) не поддерживаются строгой версией языка. Версия языка указывается в HTML-документе с помощью определения типа документа (doctype):

– Строгий (*Strict*) - не содержит элементов, помеченных как «устаревшие» или «нерекомендованные»;

– Переходный (*Transitional*) - содержит устаревшие теги в целях совместимости и упрощения перехода со старых версий HTML;

– С фреймами (*Frameset*) - аналогичен переходному, но содержит также теги для создания фреймовых структур.

Начиная с 2004 года W3C вел разработку HTML версии 5. Черновой вариант спецификации языка появился в Интернете 20 ноября 2007 года.

В HTML5 появляется множество синтаксических особенностей. HTML5 вводит несколько новых элементов и атрибутов, некоторые устаревшие элементы, которые еще можно было использовать в HTML 4.01 (такие как font и center), были исключены, исключена и поддержка фреймов.

На рабочем семинаре в 1998 г. консорциум W3C решил, что с точки зрения языков разметки будущим Web является развитие HTML в сторону обеспечения совместимости с XML. Поэтому W3C подвел черту под HTML 4.01 и сконцентрировался после этого на спецификации XHTML 1.0 (*Extensible Hypertext Markup Language* - расширяемый язык разметки гипертекста), законченной в начале 2000 г. XHTML по своим возможностям сопоставим с HTML, однако предъявляет более строгие требования к синтаксису.

Вскоре последовал язык XHTML 2.0, который добавил целый пакет новых мощных средств с целью стать следующей основой Web. Проблема с XHTML 2.0 состояла в том, что он не был обратно совместимым с уже имеющейся Web-разметкой.

HTML5 - это попытка определить единый язык разметки, который мог бы способствовать созданию нового поколения Web-приложений, не разрушая - что критически важно - обратной совместимости как с HTML, так и с XHTML.

2.2. Основы языка HTML

Структура документа HTML

Основными конструкциями языка являются тэги. Все тэги HTML начинаются с "<" (левой угловой скобки) и заканчиваются символом ">" (правой угловой скобки). Как правило, существует стартовый тэг и завершающий тэг.

<title> Заголовок документа </title>

Завершающий тэг выглядит так же, как стартовый, он отличается от

него прямым слэшем перед текстом внутри угловых скобок. Некоторые тэги, такие как <hr> (тэг, определяющий горизонтальную линию), не требуют завершающего тэга.

HTML не реагирует на регистр символов, на синтаксис. Тэги либо распознаются браузером, либо нет.

Когда Web-браузер получает документ, он определяет, как документ должен быть интерпретирован. Самый первый тэг, который встречается в документе, должен быть тэгом <html>. Простейший HTML-документ будет выглядеть так:

```
<html>
  <head>
        <title>.....</title>
  </head>
  <body>
        ……..
  </body>
</html>
```

Основные элементы страницы

Заголовочная часть документа <head> .

Элементы, находящиеся внутри раздела head (кроме названия документа, записываемого с помощью раздела title), не видны на экране (во всяком случае, напрямую). Элементы, содержащиеся внутри раздела head документа, нужны для того, чтобы:

– дать документу название;

– определить отношения между несколькими документами;

– дать указание браузеру создать форму для поиска;

– добавить динамическую составляющую.

Стартовый тэг <head> помещается непосредственно перед тэгом <title>, а завершающий тэг </head> размещается сразу после окончания описания документа. Например:

```
<head>
<title> Список сотрудников </title>
</head>
```

Заголовок документа <title> .

Раздел title служит для того, чтобы дать документу название. Оно обычно показывается в заголовке окна браузера.

Название документа записывается между тэгами <title> и </title> и представляет собой текстовую строку.

Тело документа <body> .

Тело документа должно находиться между тэгами <body> и </body>. Атрибуты этого тега определяют общий облик вашего документа. Они перечислены в табл.3.

Таблица 3. Атрибуты тега <body>

Атрибут	Назначение
background	Указывает на URL-адрес изображения, которое используется в качестве фонового.
bgcolor	Определяет цвет фона документа.
bgproperties	Если установлено значение fixed, фоновое изображение не прокручивается.
alink	Определяет цвет активной ссылки.
link	Определяет цвет еще не просмотренной ссылки.
vlink	Определяет цвет уже просмотренной ссылки.
text	Определяет цвет текста.
topmargin	Устанавливает границу верхнего поля в пикселах.
leftmargin	Устанавливает границу левого поля в пикселах.

Комментарии

Как любой язык, HTML позволяет вставлять в тело документа комментарии, которые сохраняются при передаче документа по сети, но не отображаются браузером. Синтаксис комментария:

<!-- Это комментарий -->

Комментарии могут встречаться в документе где угодно и в любом количестве.

Элемент address

Этот элемент служит для идентификации автора документа. Сюда же обычно помещаются сведения об авторских правах. Этот элемент располагается либо в начале, либо в самом конце документа. Элемент address состоит из текста, помещенного между тэгами <address> и </address>. Текст, заключенный между этими тэгами, обычно показывается в окне браузера курсивом.

Цветовое оформление документа

Если цвета составных частей документа не определены автором, то используются цвета по умолчанию. Они определяются установками программы просмотра. В HTML цвета определяются цифрами в шестнадцатеричном коде. Цветовая система базируется на трех основных цветах – красном, зеленом и синем – и обозначается RGB. Для каждого цвета задается шестнадцатеричное значение в пределах от 00 до FF, что соответствует диапазону 0–255 в десятичном исчислении. Затем эти значения объединяются в одно число, перед которым ставится символ #. Например, число #800080 обозначает фиолетовый цвет. Для простоты в HTML определены 16 стандартных цветов, которые вместе с их шестнадцатеричными кодами приведены в табл. 4.

Таблица 4. Шестнадцатеричные коды 16 стандартных цветов

Цвет	Код	Цвет	Код
Black (черный)	#000000	Silver (серебряный)	#C0C0C0
Maroon (темно-бордовый)	#800000	Red (красный)	#FF0000
Green (зеленый)	#008000	Lime (известь)	#00FF00
Olive (оливковый)	#808000	Yellow (желтый)	#FFFF00
Navy (темно-синий)	#000080	Blue (синий)	#0000FF
Purple (фиолетовый)	#800080	Fuchsia (фуксия)	#FF00FF
Teal (чирок)	#008080	Aqua (аква)	#00FFFF
Gray (серый)	#808080	White (белый)	#FFFFFF

Атрибут bgcolor отвечает за цвет фона документа. Атрибут text определяет цвет текста документа. Атрибут link используется браузером для показа ещё непросмотренных ссылок. Атрибут vlink служит для показа уже просмотренных ссылок. Как правило, их окрашивают более темным оттенком того же цвета, что и непросмотренные ссылки. Атрибут alink определяет цвет ссылки, активной в текущий момент.

Установка полей

Атрибут leftmargin устанавливает расстояние между левым краем текста и левым краем окна браузера, которое измеряется в пикселах. Атрибут topmargin служит для установки расстояния между верхним краем текста и верхним краем окна браузера.

Специальные символы

Некоторые специальные символы не входят в базовую часть таблицы кодов ASCII. К ним относятся буквы алфавитов части европейских языков, математические и некоторые другие символы. Однако они тоже могут быть введены в ваш HTML-документ при помощи символа & (амперсант) и имени символа. Например, пробел: .

Форматирование документов

Любые опубликованные материалы имеют определенную структуру.

Разделение текста на абзацы

Поместите открывающий тэг в начало каждого нового абзаца вашего текста, и программа просмотра отделит абзацы друг от друга пустой строкой. Использование закрывающего тэга необязательно.

Атрибут align, имеющий значения, представлен в табл. 5. По умолчанию происходит выравнивание по левому краю.

Таблица 5. Значения атрибута align

Значение	**Функция**
left	Выравнивание текста по левой границе окна браузера.
center	Выравнивание по центру окна браузера.
right	Выравнивание по правой границе окна браузера.

В HTML несколько стоящих подряд тэгов <p> не дают дополнительного пространства между абзацами.

Перевод строки

Для того чтобы перейти на следующую строку в любом нужном вам месте текущей строки, в HTML существует тэг разрыва строки
. Он заставляет программу просмотра выводить стоящие после него символы с начала новой строки. В отличие от тэга абзаца тэг
 не добавляет пустую строку.

Пример:

<body>
Эта и следующая строка разделены тэгом конца абзаца
<p>
А эти две строки разделены тэгом конца строки.

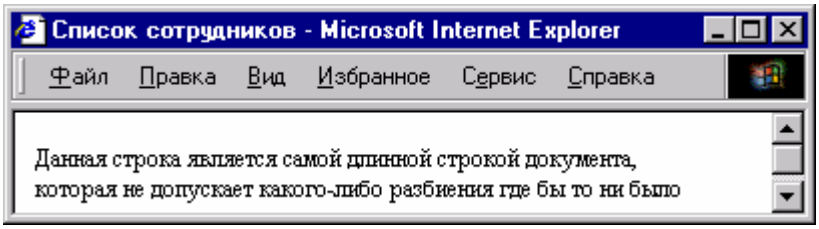
Разница видна?
</body>

Бывают случаи, когда возникает надобность в операции противоположного назначения – запретить перевод строки. Текст, заключенный между тэгами <nobr> и </nobr>, будет гарантированно располагаться в одной строке без переноса на другую.

Во избежание неприятностей с элементом <nobr> вы можете предложить браузеру читателя альтернативное место перевода строки при помощи тэга <wbr> ("мягкий" перевод строки). Эта инструкция будет выполнена только в том случае, если браузер не сможет вывести вашу фразу одной строкой в пределах окна просмотра.

Пример:

<nobr> Данная строка является самой длинной строкой документа,<wbr> которая не допускает какого-либо разбиения где бы то ни было </nobr>



В данном случае перевод строки будет произведен только после запятой.

Структурирование текста

Для удобства читателей текст рекомендуется разбить на логические части, каждая из которых посвящена отдельной теме.

Заголовки

Элемент «заголовок» является контейнером и поэтому должен иметь открывающий (<h1>) и закрывающий (</h1>) тэги. HTML располагает шестью уровнями заголовков: h1 (самый верхний), h2, h3, h4, h5 и h6 (самый нижний). Программа просмотра выводит каждый из них, но вы не можете точно знать, в каком именно виде. Это обстоятельство является частью философии HTML: вы как автор отвечаете за содержание, а читатель

определяет окончательный облик документа. Предназначение заголовков – показать структуру вашего документа. Точно так же, как и в теге «абзац», в заголовках можно использовать атрибут align.

Горизонтальные линии

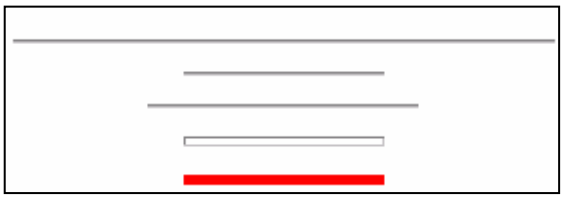
Элемент <hr> позволяет провести рельефную горизонтальную линию в окне большинства программ просмотра. Этот тэг не является контейнером, поэтому не требует закрывающего тэга. До и после линии автоматически вставляется пустая строка.

Таблица 6. Атрибуты элемента <hr>

Атрибут	Назначение
align	Выравнивает по краю или центру; имеет значения left, center, right.
width	Устанавливает длину линии в пикселах или процентах от ширины окна браузера; в последнем случае добавляется символ %.
size	Устанавливает ширину линии в пикселах.
noshade	Отменяет рельефность линии.
color	Указывает цвет линии. Используется формат RGB или стандартное имя.

Пример:

```
<body>
<hr align=center>
< hr align=center width=100>
< hr align=center width=50%>
< hr align=center width =100 size=5>
< hr align=center width =100 size=5 color=red>
```



```
</body>
```

Использование предварительно отформатированного текста

Наиболее употребляемым является контейнер <pre>. Текст внутри него может записываться в любой форме. Поддерживаются также тэги <p> и
. Универсальность этого контейнера позволяет создавать таблицы и ровные колонки текста.

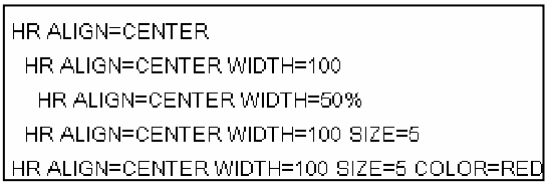
Текст внутри контейнера <pre> может содержать любые элементы физического и логического форматирования. Однако запрещено использование тэга <address> и тэгов заголовка.

Самым большим недостатком контейнера <pre> является возможность вывода преформатированного текста только моноширинным шрифтом.

Пример:

```
<pre>
 HR ALIGN=CENTER
   HR ALIGN=CENTER WIDTH=100
     HR ALIGN=CENTER WIDTH=50%
   HR ALIGN=CENTER WIDTH=100 SIZE=5
 HR ALIGN=CENTER WIDTH=100 SIZE=5 COLOR=RED
```

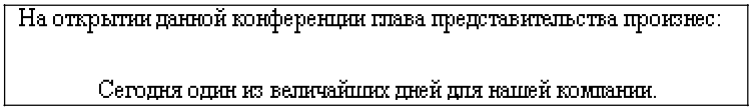
</pre>



Цитата <blockquote>

Данный тэг предназначен для обозначения в документе цитаты из другого источника. Текст, обозначенный тэгом <blockquote>, отступает от левого края документа на 8 пробелов.

<body>
На открытии данной конференции глава представительства произнес:
<blockquote>
Сегодня один из величайших дней для нашей компании.
</ blockquote>
</body>
При отображении браузером текст будет выглядеть так:



Форматирование текста

Логическое форматирование

Таблица 7. Элементы логического форматирования

<cite>	Используется для выделения цитат или названий книг и статей, при этом текст обычно выводится курсивом <cite>Tom Sawyer</cite> remains one of the classics of American literature.
<code>	Применяется для вывода небольшого куска программного кода (для больших листингов используется тэг <pre>) шрифтом фиксированной ширины.
	Этот элемент обычно используется для выделения важных фрагментов текста. Браузеры обычно отображают такой текст курсивом. The actual line reads, "Alas, poor Yorick. I knew him, Horatio."
<kbd>	Элемент, выделяющий шрифтом фиксированной ширины текст, вводимый пользователем с клавиатуры. To run the decoder, type <kbd>Restore</kbd> followed by your password.
<samp>	Используется для выделения нескольких символов шрифтом фиксированной ширины. The letters <samp>AEIOU</samp> are the vowels of the English language.
	Этот элемент обычно используется для выделения важных фрагментов текста. Браузеры обычно отображают такой текст полужирным шрифтом. The most important rule to remember is Don't panic!
<var>	Используется для отметки имен переменных. Обычно такой текст отображается курсивом. The sort routine rotates on the <var>I</var>th element.

Все перечисленные элементы являются контейнерами и требуют закрывающего тэга.

Физическое форматирование

Последней «инстанцией» определения внешнего вида документа

является программа просмотра читателя. Вы имеете ограниченные возможности повлиять на этот процесс с помощью элементов физического форматирования, список которых приведен ниже. Физическое форматирование считается «не рекомендованным» для оформления Web-страниц.

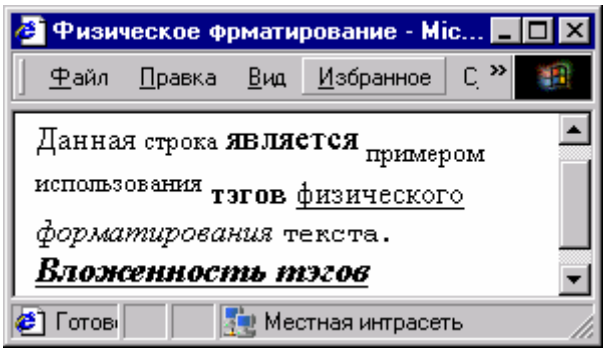
В последних версиях языка вместо него используются каскадные таблицы стилей. Но тем не менее тэги физического форматирования можно встретить еще во многих документах.

Таблица 8. Тэги физического форматирования

	Выделяет текст полужирным шрифтом
	This is in bold text
	Выделяет текст курсивом
	This is initalic text
<tt>	Выводит текст шрифтом фиксированной ширины
	This is in <tt>teletype</tt> text
<u>	Элемент подчёркивания
	This text is <u>underlined</u>
<strike>	Элемент зачеркивания. Отображается текст, перечеркнутый горизонтальной линией
	This is a <strike>strikethough</strike> example
<big>	Выводит текст шрифтом большего размера
	This is <big>big</big> text
<small>	Выводит текст шрифтом меньшего размера
	This is small text
<sub>	Сдвигает текст ниже уровня строки и выводит его (если возможно) шрифтом меньшего размера
	This is a _{subscript}
<sup>	Сдвигает текст выше уровня строки и выводит его (если возможно) шрифтом меньшего размера
	This is a ^{superscript}

Все элементы физического форматирования являются контейнерами, т.е. требуют закрывающего тэга. Элементы физического форматирования могут быть вложенными друг в друга.

```
<body>
Данная
<small>строка</small>
<b><big>является</big></b>
<sub> примером</sub>
<sup>использования</sup>
<b>тэгов</b>
<u> физического</u>
<i> форматирования </i>
<tt>текста.</tt>
<b> <i>  <u>
<big>Вложенность тэгов</big>
</u> </i> </u>
</body>
```



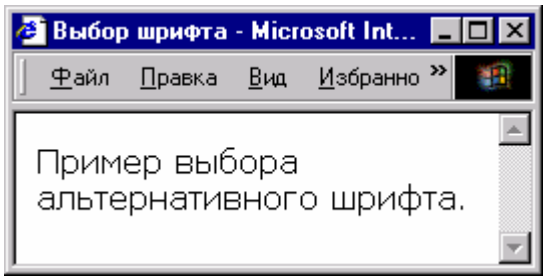
Шрифты

Элемент *font* представляет собой контейнер, т.е. требует как открывающего, так и закрывающего тэгов. После стартового тэга обязательно указание атрибутов, без которых элемент не влияет на текст, помещенный в контейнер. Элемент font может использоваться внутри любого другого текстового контейнера.

Атрибут *face* данного элемента позволяет вам указать тип шрифта, которым программа просмотра выведет ваш текст. Параметром атрибута служит название шрифта, которое должно в точности совпадать с названием шрифта, имеющегося у пользователя. Если нужного шрифта нет, программа проигнорирует запрос и будет использовать шрифт, установленный по умолчанию.

Атрибут *face* позволяет указать как один, так и несколько шрифтов (через запятую). Весь список будет просмотрен слева направо, и первый из имеющихся на машине пользователя будет использован для вывода документа.

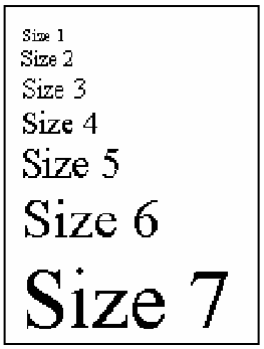
```
<html>
<head>
<title>Выбор шрифта</title>
</head>
<body>
<font face="Verdana", "Arial", "Helvetica">
Пример выбора альтернативного шрифта. </font>
</body>
</html>
```



Атрибут *size* служит для указания размера шрифта в условных единицах от 1 до 7. Принято считать, что размер «нормального» шрифта соответствует числу 3. Размер может быть указан как абсолютной величиной (size=5), так и относительной (size=+2).

```
<body>
<font size=1>Size 1</font><br>
```

```
<font size=-1>Size 2</font><br>
<font size=3>Size 3</font><br>
<font size=4>Size 4</font><br>
<font size=+2>Size 5</font><br>
<font size=6>Size 6</font><br>
<font size=+4>Size 7</font><br>
</body>
```



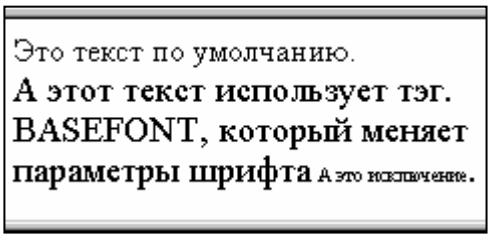
Атрибут color устанавливает цвет шрифта, который может быть указан как в формате RGB, так и стандартным именем.

```
<body>
<font color="#FF0000">Этот текст будет выведен красным цветом</FONT><BR>
<font color="green"> Этот текст будет выведен зеленым цветом
                                        </FONT><BR>
</body>
```

Тэг <basefont> служит для указания размера, типа и цвета шрифта, стандартных для данного документа. Эти величины обязательны для всего документа, если только не переназначаются при помощи элемента font. После закрытия элемента font значения тэга <basefont> восстанавливаются. Значения атрибутов basefont могут быть изменены другим тэгом <basefont> в любом месте документа. Это не контейнер, и закрывающего тэга не существует.

Тэг <basefont> использует те же самые атрибуты, что и элемент font.

```
<body>
Это текст по умолчанию.<br>
<basefont size=4 face="georgia">
А этот текст использует тэг.<br>
BASEFONT, который меняет
параметры шрифта
<font size=-3> А это
исключение</font>.<br>
</body>
```



Списки

HTML имеет специальные элементы-контейнеры для представления данных в виде списков. Основными типами списков являются нумерованные и маркированные списки, списки определений. Для получения дополнительных эффектов различные типы списков могут вкладываться друг в друга.

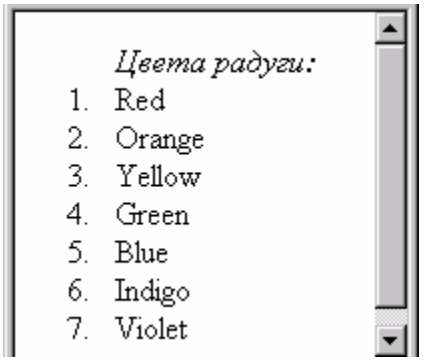
Упорядоченный (нумерованный) список

В HTML список состоит из тэга-контейнера списка, определяющего его тип, и стандартных тэгов <li>, предваряющих каждый пункт списка. Упорядоченный список используется для нумерованного перечисления отдельных пунктов или указания последовательности каких-то действий. Когда программа просмотра встречает тэг упорядоченного списка, она последовательно нумерует пункты списка: 1, 2, 3 и т.д.

Упорядоченный список открывается тэгом <ol>, а каждый его пункт начинается стандартным тэгом <li>. Для создания заголовка списка используется специальный тэг <lh>. Список закрывается тэгом . Открывающий и закрывающий тэги обеспечивают перевод строки до и после списка, отделяя, таким образом, список от остального текста.

Тэги абзаца можно использовать для отделения пунктов списка друг от друга. Внутри списка можно применять тэги стилей (как физические, так и логические, например или <i>) и другие элементы HTML.

```
<body>
<ol>
<lh><em>Цвета радуги:</em><br>
<li>Red
<li>Orange
<li>Yellow
<li>Green
<li>Blue
<li>Indigo
<li>Violet
</ol>
</body>
```



Упорядоченные списки допускают вложения друг в друга. Атрибуты тэга <ol> позволяют устанавливать вид маркеров элементов списка, а также задавать начальный маркер списка. В табл. 9 приведены атрибуты этого тэга и их назначение.

Таблица 9. Атрибуты тэга

Атрибут	Назначение
compact	Представляет список в более компактном виде.
type=A	Устанавливает маркер в виде прописных букв.
type=a	Устанавливает маркер в виде строчных букв.
type=I	Устанавливает маркер в виде больших римских цифр.
type=i	Устанавливает маркер в виде маленьких римских цифр.
type=1	Устанавливает маркер в виде арабских цифр.
start=n	Устанавливает начальный маркер в текущем списке.

Неупорядоченный (маркированный) список

В HTML существует возможность создания неупорядоченных списков, т.е. таких, в которых отношения между пунктами не определены. (Списки такого типа называют также ненумерованными или маркированными.)

Неупорядоченный список вместо буквенной или цифровой нумерации предполагает использование различных символов, называемых маркерами списка (bullets). Как и в упорядоченных списках, здесь также обеспечивается перевод строки до и после списка, а также допускается вложенность списков. Список располагается внутри контейнера <ul>. Программы просмотра создают автоматический отступ для вложенных списков и сами разнообразят маркеры, вид которых зависит от типа программы.

Как и в случае тэга <ol>, для тэга <ul> в HTML можно устанавливать вид маркеров в неупорядоченных списках при помощи атрибута type, который допускает три значения: disc, square и circle.

Как и тэг <ol>, тэг <ul> имеет атрибут compact, позволяющий выводить неупорядоченный список в более компактном виде.

Список определений

Списки определений, также называемые словарями специальных терминов (глоссариями), являются особым видом списков HTML. Они представляют текст в форме словарной статьи, состоящей из определяемого термина и абзаца, раскрывающего его значение. Элемент списка определений dl служит контейнером и обеспечивает отделение списка от остального текста пустыми строками. Внутри контейнера тэгом <dt> помечается определяемый термин, а тэгом <dd> – абзац с его определением. Тэги <dt> и <dd> не являются контейнеры и поэтому не имеют закрывающих тэгов. Базовый шаблон списка определений выглядит следующим образом:

```
<dl>
<dt>Термин
<dd>Определение данного термина
</dl>
```

Списки определений могут включать другие элементы HTML. Часто применяются элементы стилей (физические и логические).

Комбинирование списков

Бывают ситуации, когда в список одного типа требуется включить

список (или списки) другого типа. HTML позволяет производить любое комбинирование типов списков.

```
<body>
<ol>
<lh><em>Planets of the Solar System:</em><br>
<li>Mercury
<ul> <ul>
<li>Roman god of commerce, travel, and thievery
<li>Dictionary Definition
<dl>
<dt>Mercury
<dd>The smallest of the planets and the one nearest
the sun, having a sidereal period of revolution about
the sun of 88.0 days at a mean distance of 58.3
million kilometers (36.2 million miles) and a mean
radius of appropriately 2,414 kilometers (1,500
miles).
</dl>
</ul> </ul>
<li>Venus
<ul> <ul>
<li>Roman goddess of sexual love and physical beauty
<li>Dictionary Definition
<dl>
<dt>Venus
 <dd>The second planet from the sun, having an average radius of 6,052
kilometers  (3,760 miles), a mass 0.815 times that of Earth, anda sidereal
period of revolution about the sun of 224.7 days at a mean distance of
approximately 100.1 million kilometers (67.2 million miles).
</dl>
</ul> </ul>
</ol>
</body>
```

Planets of the Solar System:

1. Mercury
   - Roman god of commerce, travel, and thievery
   - Dictionary Definition

   Mercury

   The smallest of the planets and the one nearest the sun, having a sidereal period of revolution about the sun of 88.0 days at a mean distance of 58.3 million kilometers (36.2 million miles) and a mean radius of appropriately 2,414 kilometers (1,500 miles).
2. Venus
   - Roman goddess of sexual love and physical beauty
   - Dictionary Definition

   Venus

   The second planet from the sun, having an average radius of 6,052 kilometers (3,760 miles), a mass 0.815 times that of Earth, anda sidereal period of revolution about the sun of 224.7 days at a mean distance of approximately 100.1 million kilometers (67.2 million miles).

Отступ для каждого списка устанавливает программа просмотра, однако, если это необходимо, для его увеличения можно добавить «пустой» список:

Простой список
 Простой список

нужно написать

Список с увеличенным отступом
 Список с увеличенным отступом

1. Простой список
2. Простой список

    1. Список с увеличенным отступом
    2. Список с увеличенным отступом

Дополнительные возможности форматирования списков

Вы можете создать собственные маркеры для использования в ненумерованных списках.

Контейнер ul информирует браузер о необходимости интерпретировать заключенный в нем текст как неупорядоченный список. Если не нужны стандартные маркеры, то тэги <li> не используется. Вместо него нужно записать код, определяющий ваш новый маркер, например:

<img src= "cube.gif" align=top>Red

Тэг <img> указывает на графический файл используемого маркера и метод выравнивания изображения.

Гиперссылки

Значение ссылок во Всемирной паутине трудно переоценить. Читая книгу, вы всегда имеете ее под рукой. Работая в WWW, вы понятия не имеете, где находится та или иная нужная вам страница. Поэтому ссылки здесь являются единственной возможностью перейти от одного документа к другому.

Гипертекст и гипермедиа

Гипертекстовый документ – это документ, содержащий ссылки на другие документы, позволяющие при помощи нажатия кнопки мыши быстро перемещаться от одного документа к другому.

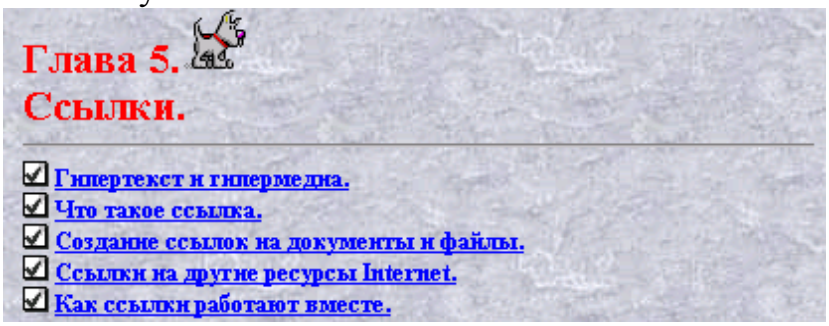
Гипермедийный документ основан на гипертекстовом документе, но в дополнение к тексту содержит разнообразную графику, видео- и аудиообъекты. В таких документах в качестве ссылок часто используются изображения.

Ссылка состоит из двух частей. Первая из них – это то, что вы видите на Web-странице; она называется указатель (anchor). Вторая часть, дающая инструкцию браузеру, называется адресной частью ссылки (URL-адресом). Когда вы щелкаете мышью по указателю ссылки, браузер загружает документ, адрес которого определяется URL-адресом.

Указатели

Указатели бывают двух типов – текстовые и графические. Текстовые указатели представляют собой слово или слова, подчеркнутые прямой линией.

Ниже приведен пример Web-страницы, содержащей текстовые и графические указатели.



Графические указатели в принципе мало отличаются от текстовых. Они не подчеркиваются и не выделяются цветом, но вокруг них можно сделать рамку.

Маркеры списка (Bullets)

Графические указатели часто имеют вид точки, звездочки или другого маркера. Как правило, строка рядом с маркером также служит указателем, имеющим тот же самый URL-адрес.

Второй частью ссылки является URL-адрес. Это не что иное, как адрес Web-страницы, которая будет загружена, если щелкнуть на указателе ссылки кнопкой мыши. Указание адреса может быть относительным или абсолютным.

Относительные указатели

URL-адрес файла, расположенный на том же компьютере, что и документ, в котором находится указатель этой ссылки, называется относительным. Это означает, что если браузер загрузил страницу, находящуюся по адресу http://www.mysite.com/page, то относительный адрес /picture подразумевает адрес http://www.mysite.com/page/picture, т.е. подкаталог, расположенный на той же машине.

Абсолютные указатели

URL-адрес, полностью определяющий компьютер, каталог и файл, называется абсолютным. В отличие от относительных адресов абсолютные адреса могут ссылаться на файлы, расположенные на других компьютерах.

Создание ссылок на документы и файлы

Для создания ссылки необходимо сообщить браузеру, какой элемент страницы является указателем, и указать адрес документа, на который ссылаетесь. Оба действия выполняются при помощи тэга <a>.

Тэг <a> имеет атрибут href, в котором размещается URL-адрес.

anchor

Для облегчения работы с относительными указателями в HTML введен тэг <base>. Он располагается в начале документа и содержит URL-адрес, относительно которого в документе построена вся адресация. Это указание влияет на любой нижестоящий тэг, включая <a>, <img> и т.д. Если, например, вы вставите строку <base href="http://www.server.com">, вся относительная адресация в дальнейшем будет базироваться на этом адресе.

Так, относительный указатель на файл "images/face.gif будет подразумевать адрес http://www.server.com/images/face.gif. Если тэг <base> отсутствует, относительная адресация строится на URL-адресе текущего документа.

Некоторые программы просмотра выводят в маленьком окошке текст, содержащийся в атрибуте title тэга <a>, если задержать курсор мыши на указателе ссылки.

 Me Page

Создание графического указателя аналогично созданию текстового указателя. Вместо текста в тэге <a> нужно разместить имя файла изображения. В следующей строке кода атрибут href опять указывает адрес домашней страницы, только вместо текстового анкера при помощи тэга <img> создается графический указатель, представляющий собой маленькую каринку

<img src="comp.jpg">

Кроме ссылок на другие документы, часто бывает полезно включить ссылки на разные части текущего документа.

Для построения внутренней ссылки сначала нужно создать указатель, показывающий место назначения. Для этого нужно разместить там указатель и дать ему имя при помощи атрибута name тэга <a>. При этом атрибут href не используется, и браузер не выделяет содержимое тэга <a>.

Middle Section in Web page

После того как указатель получил имя, можно приступить к созданию ссылки на него. Для этого вместо указания в атрибуте href адреса документа, как это делалось ранее, поместим туда имя указателя со специальным префиксом (#), говорящим о том, что это внутренняя ссылка.

Jump to the middle

Теперь, если пользователь щёлкнет кнопкой мыши на словах Jump to the middle, программа просмотра выведет среднюю часть документа, причем указатель ссылки будет расположен в верхней строке окна.

Файлы и другие встраиваемые объекты

Когда пользователь щелкает мышью на ссылке, указывающей на другую Web-страницу, она выводится непосредственно в окне браузера. Если же ссылка указывает на документ иного типа, программа просмотра принимает документ и затем решает, что с ним делать потом.

Связь с электронной почтой (e-mail).

Если вас интересует отклик читателей на содержание вашего документа, вы захотите поместить на странице свой адрес e-mail.

Send me E-mail.

После щелчка мышью на ссылке на ваш адрес браузер откроет собственное окно для работы с электронной почтой.

Связь с FTP

Ссылка на сайт FTP похожа на другие гипертекстовые ссылки. Вместо http: нужно поставить ftp:, а вместо URL-адреса – //sitename/path. Вы должны

удостовериться, что сайт, на который вы ссылаетесь, разрешает анонимный доступ. Практически все браузеры работают с FTP без всяких проблем. Ваша ссылка может иметь следующий вид:

You can get the FAQ <a href="ftp://ftp.mysite.com/pub/FAQ">here< /a>.

Если не указывать имя файла, браузер выведет перечень всех файлов в каталоге.

Таблица десять. Связь с FTP

Ссылка на	Формат	Пример
Web-страницу	http://sitename/	http://www.mysite.com/
e-mail	mailto:address	mailto:me@mysite.com
Newsgroup	news:newsgroupname	news:news.newusers.questions
FTP	ftp://sitename/	ftp://ftp.mysite.com/
Telnet	telnet://sitename/	telnet://bbs.mysite.com/

Таблицы

Для лучшего представления информации вы можете использовать таблицы. Элемент table представляет собой тэг-контейнер, в котором размещается содержимое таблицы.

Таблица строится по строкам: для обозначения строки используется контейнер tr, для обозначения заголовков столбцов (строк) – контейнер th, а для данных в ячейках – контейнер td. Заголовки выделяются полужирным шрифтом и центрируются в своих ячейках. Данные имеют обычный шрифт и выравниваются по левой стороне ячейки.

Таблица 11. Атрибуты элемента <table>

Тэг	Описание
<table></table>	Контейнер таблицы.
<tr></tr>	Контейнер строки таблицы. Допускается отсутствие закрывающего тэга.
<td></td>	Контейнер ячейки таблицы. В ячейку можно включить другую таблицу. Закрывающий тэг может быть опущен.
<th></th>	Контейнер заголовка, располагающегося обычно в первой строке, либо в первом столбце таблицы. Закрывающий тэг также необязателен.

Пример:

```
<table border=2>
<tr>
<th>Colors</th><th>Of</th><th>The Rainbow</th>
</tr>
<tr>
<td>Red</td><td>Orange</td><td>Yellow</td>
</tr>
<tr>
<td>Green</td><td>Blue</td><td>Violet</td>
</tr>
</table>
<hr>
<table border=2>
```

```
<caption>My Favorite Groups</caption>
<tr><th>Rock</th>
<td>Pink Floyd</td>
<td>Led Zepplin</td>
<td>The Dobbie Brothers</td></tr>
<tr><th>Soft</th><td>Simon and Garfunkel</td>
<td>Peter, paul, & Mary</td>
<td>Neil Young</td></tr>
<tr><th>New Age</th><td>Enya</td>
<td>Clannad</td>
<td>Steamroller</td></tr>
</table>
```

Colors	Of	The Rainbow
Red	Orange	Yellow
Green	Blue	Violet

My Favorite Groups

Rock	Pink Floyd	Led Zepplin	The Dobbie Brothers
Soft	Simon and Garfunkel	Peter, paul, & Mary	Neil Young
New Age	Enya	Clannad	Steamroller

Если надо разнообразить заголовки, то можно применить тэги физического форматирования.

Таблицы всегда должны быть прямоугольными. Другие формы не допускаются.

Размещение данных внутри ячеек

При помощи атрибутов align и valign можно по-разному размещать данные относительно границ ячейки. Эти атрибуты используются совместно с элементами <caption>, <tr>, <TH> и <td> в самых различных комбинациях. В табл. 12 приведены значения атрибутов для перечисленных элементов.

Таблица 12. Теги атрибутов элементов <caption>, <tr>, <TH> и <td>

<caption>	Атрибут align может иметь значения top и bottom (по умолчанию – top); размещает заголовок таблицы сверху или снизу.
<tr>	Атрибут align может принимать значения left, center и right (по умолчанию – left для данных и center для заголовков); он определяет горизонтальное выравнивание данных в ячейках и действует на всю строку, если не отменяется тем же атрибутом в отдельной ячейке. Атрибут valing может иметь значения top, bottom, middle и baseline (по умолчанию – middle); он регулирует положение данных относительно верхней и нижней границ ячейки и влияет на всю строку, если не отменяется таким же атрибутом в отдельной ячейке, baseline применяется ко всем элементам строки и выравнивает их по базовой линии.
<th>	Атрибут align может принимать значения left, center и right (по умолчанию – center). Атрибут valing может иметь значения top, bottom и middle (по умолчанию – middle).
<td>	Атрибут align может принимать значения left, center и right (по умолчанию – left). Атрибуту valing может иметь значения top, bottom и middle (по умолчанию – middle).

Применение этих атрибутов:

```
< table border>
<caption align=bottom>A Really Ugly Table</caption>
<tr>
<th></th><th>##########</th><th>##########</th>
<th>##########</th>
</tr>
<tr align=right>
<th>Row 1</th><td>XX<br>XX</td><td align=center>X
</td><td>XXX</td>
</tr>
<tr valing=baseline>
<th aling=left>Second Row</th><td>XXX<br>XXX</td><td>XXX</td>
<td>XXX<br>XXXXX<br>XXX</td>
</tr>
<tr aling=left>
<th>This Is<br>The Bottom Row of <br>The Table</th>
<td valing=bottom>XXXXX</td>
<td valing=top>XXX<br>XXXXX</td>
<td valing=middle>XXXXX</td>
</tr>
</table>
```

	##########	##########	##########
Row 1	XX XX	X	XXX
Second Row	XXX XXX	XXX	XXX XXXXX XXX
This Is The Bottom Row of The Table	XXXXX	XXX XXXXX	XXXXX

A Really Ugly Table

.

При отсутствии атрибута border рамка не прорисовывается. Такие таблицы можно использовать для табличной верстки старниц.

Объединение ячеек

Смежные ячейки таблицы могут объединяться с целью размещения большего количества данных. Например, в таблице из пяти строк и пяти столбцов все ячейки первой строки можно объединить и поместить в этой строке красивый заголовок таблицы. Возможно также объединение нескольких строк или создание пустой прямоугольной области.

Для соединения двух смежных ячеек в одном столбце нужно использовать атрибут rowspan тэга <th> или <td>, например: <td rowspan=2>

Для объединения двух смежных ячеек в одной строке нужно использовать атрибут colspan тех же тэгов, например: <td colspan=2>

Пустые ячейки

Существует разница между пустой ячейкой и ячейкой с невидимым содержанием

Таблица 13. Атрибуты по работе с таблицей

width	Определяет ширину всей таблицы в пикселах, либо в процентах от ширины окна браузера. Может также использоваться для отдельной ячейки.
height	Определяет высоту всей таблицы в пикселах, либо в процентах от высоты окна браузера. Может также использоваться дня отдельной ячейки.
border	Устанавливает ширину рамки таблицы в пикселях, например, border=2.
cellpading cellspasing	Добавляют свободное пространство между данными внутри ячейки и ее границами и, соответственно, между ячейками внутри всей таблицы. Если рамка отсутствует, то результат их действия одинаков.

Использование цветов

Вы можете изменить цвет фона ячейки при помощи атрибута bgcolor перед размещением в ней текста или изображения, а также использовать атрибут bordercolor для изменения цвета рамки ячейки.

Тэги <table>, <td>, <th> и <tr> допускают использование в них указанных атрибутов.

Теги <table> и <tr> поддерживают аттрибут background.

Альтернативные способы представления табличных данных:

– Список.

– Использоваие изображения.

– Предварительно отформатированный текст.

<pre>

```
 +------------------+----------------------+--------------------+
 | Offense   | Defense   | Goalie  |
 +------------------+----------------------+--------------------+
| Husmann   | O'Donnell | |
| Popplewell| |
| McGilly   | Longo | Weinberg|
| Donahue   | Seymour   | |
| Camillo   | Walsh | |
+------------------+----------------------+--------------------+
```

</pre>

```
+------------------+----------------------+-----------
| Offense          | Defense              | Goalie
+------------------+----------------------+-----------
| Husmann          | O'Donnell            |
| Popplewell                              |
| McGilly          | Longo                | Weinberg
| Donahue          | Seymour              |
| Camillo          | Walsh                |
```

Использование изображения в качестве заголовка таблицы

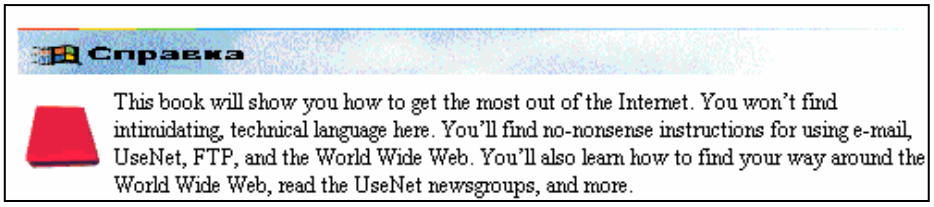
Вы можете украсить свою таблицу, поместив в ее заголовок изображение вместо текста.

<table width=500 cellspacing=0 cellpadding=2 border=0>

<tr>

<th colspan=2>

<img src="cloud.gif" border=0 height=25 width=500>
</th>
</tr>
<tr>
<td valing=top>
<img src="an59.gif">
</td>
<td valing=top>
This book will show you how to get the most out of the Internet. You won't find intimidating, technical language here. You'll find no-nonsense instructions for using e-mail,
UseNet, FTP, and the World Wide Web. You'll also learn how to find your way around the World Wide Web, read the UseNet newsgroups, and more.
</td>
</tr>
</table>



Использование графики

Изображения могут сделать текст вашего документа более содержательным.

Изображения помогают лучше передать суть и содержание документа.

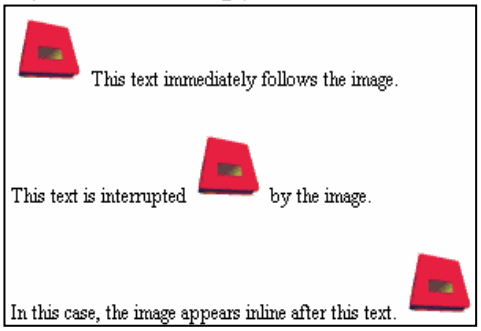
Вставка изображения в документ

Для вставки изображения на страницу следует воспользоваться тэгом <img> совместно с атрибутом src, поместив их в надлежащее место вашего HTML-документа:

<img src="URL">

По умолчанию браузер выводит изображение немедленно после текста или другого объекта, описанного предыдущими инструкциями

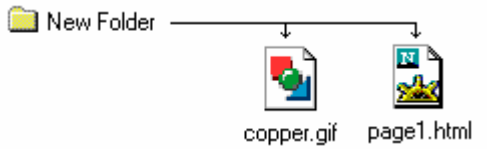
<body>
<p>
<img src="book.gif">
This text immediately follows the image.
<p>
This text is interrupted
<img src="book.gif"> by the image.
<p>
In this case, the image appears inline after this text.
<img src="book.gif">
</body>



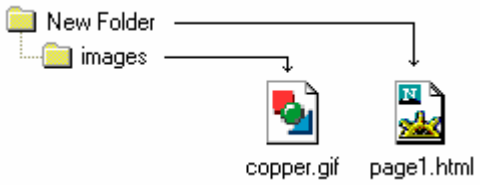
Предусмотрите хранение файлов графики в одном определенном

каталоге (не корневом) вашего Web-сайта. Тогда вы сможете использовать относительную адресацию в комбинации с тэгом <base>, а не указывать полный URL-адрес.

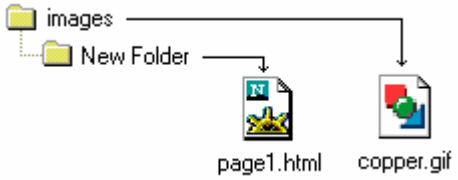
Атрибут src определяет не только какое изображение, но и где хранится это изображение. В примере, который приведен ниже. src="copper.gif" означает, что браузер будет искать изображение с именем copper.gif в той же самой папке (или каталоге), где непосредственно расположен HTML-документ.



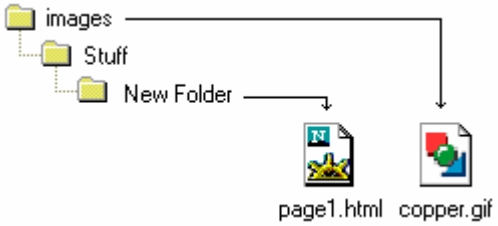
– src="images/copper.gif" означает, что изображение из папки нижнего уровня, по сравнению с папкой HTML-документа, который запросил его. Это можно продолжить по уровням вниз настолько это необходимо.



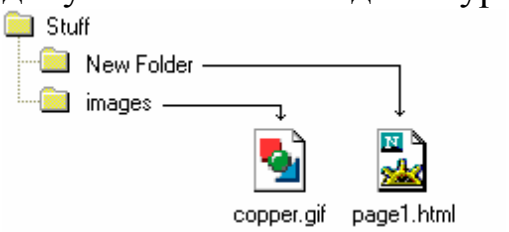
– src="../copper.gif" означает, что изображение находится в папке верхнего уровня, по сравнению с папкой HTML-документа, который запросил его.



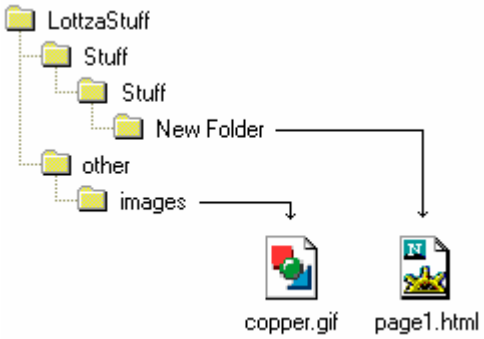
– src="../../copper.gif" означает, что изображение в папке на два уровня выше, чем папка HTML-документа, который запросил его.



– src="../images/copper.gif" означает, что изображение и HTML-документ в папках одного уровня.



– src="../../../other/images/copper.gif" - …



Выравнивание текста по краю изображения

По умолчанию, когда изображение вставляется в строку текста, строка выравнивается по низу изображения. Изменить эту установку можно при помощи атрибута align тэга <img> табл.14.

Таблица14. Атрибут align тэга <img>

Значение	Описание
top	Выравнивает текст по верху изображения.
middle	Выравнивает текст по середине изображения.
bottom	Выравнивает текст по низу изображения.

```
<body>
<p>
<img src="comp.jpg" align=top>
This text is aligned with the top of the image.
</p>
<p>
<img src="comp.jpg" align=middle>
This text is aligned with the middle of the image.
</p>
<p>
<img src="comp.jpg" align=bottom>
This text is aligned with the bottom of the image.
</p>
</body>
```

Позиционирование изображения на странице

По умолчанию программа просмотра выводит изображение в текущей строке. Текст не «обтекает» его. Однако при помощи атрибута align тэга <img> изображение можно сделать «плавающим», т. е. заставить текст расположиться вокруг изображения табл.15.

Таблица 15. Атрибут позиционирование изображения на странице

Значение	Описание
left	Обтекаемое текстом изображение прижато к левой стороне окна браузера.
right	Обтекаемое текстом изображение прижато к правой стороне окна браузера.

```
<p>
<img src="comp.jpg" align=left>
This text will wrap around the right-hand and bottom of the image. This text
```

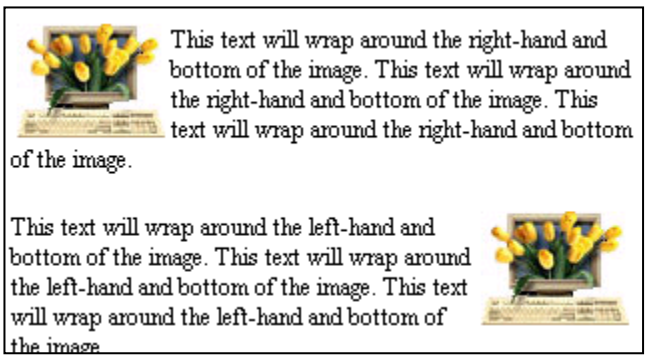
will wrap around the right-hand and bottom of the image. This text will wrap around the right-hand and bottom of the image.

<img src="comp.jpg" align=right>

This text will wrap around the left-hand and bottom of the image. This text will wrap around the left-hand and bottom of the image. This text will wrap around the left-hand and bottom of the image.

При помощи тэга <img> программе просмотра можно сообщить размеры изображения, которое затем размещено на странице:

<img src="comp.jpg">
<img src="comp.jpg" width=140 height=100>



Для указания размеров изображения (в пикселях) служат атрибуты height и width тэга <img>. Если указанные размеры не совпадают с размерами изображения, программа просмотра изменяет масштаб изображения.

Альтернативное описание изображения:

<img src="filename" alt="Description">

Если браузер читателя не выводит изображение, на его месте будет помещено альтернативное описание. Если изображение выводится, это описание будет находиться на месте иллюстрации до начала ее загрузки. Еще лучше использовать эту возможность совместно с указанием размеров.

Если указатель мыши задержать на иллюстрации на одну-две секунды, этот же текст будет выведен в специальном всплывающем окошке в виде подсказки.

Помещение изображения в рамку

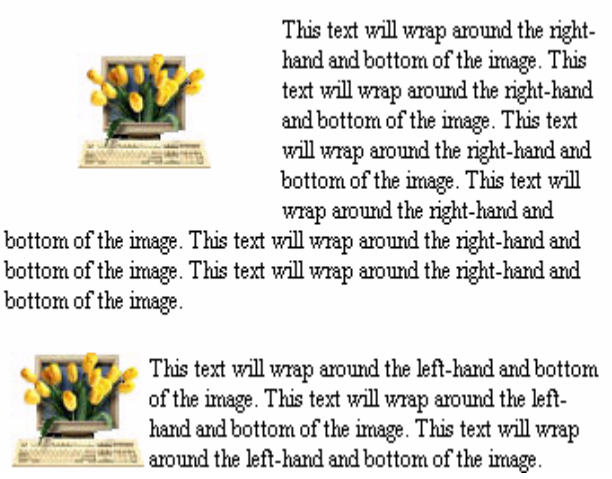
Эта простая операция предполагает применение атрибута border тэга <img>. По умолчанию программа просмотра использует рамку, которая указана в соответствующей ссылке. Введите ширину рамки в пикселах, как показано в примере:

<body>
<img src="comp.jpg" border=0
<img src="comp.jpg" border=5
<img src="comp.jpg" border=1
</body>

Отделение изображения от текста

Вам может не понравиться, что текст слишком близко подходит к изображению Если это так, используйте атрибуты vspace и hspace для указания расстояния (по вертикали и горизонтали) между кромкой текста и краями иллюстрации.

```
<body>
    <p>
<img src="comp.jpg" aling=left hspace=40 vspace=20>
This text will wrap around the right-hand and bottom of the image. This text will wrap around the right-hand and bottom of the image. This text will wrap around the right-hand and bottom of the image. This text will wrap around the right-hand and bottom of the image. This text will wrap around the right-hand and bottom of the image. This text will wrap around the right-hand and bottom of the image.
</p>
<p>
<img src="comp.jpg" align=left>
This text will wrap around the left-hand and bottom of the image. This text will wrap around the left-hand and bottom of the image. This text will wrap around the left-hand and bottom of the image.
</p>
</body>
```



Карта ссылок

Сегодня многие Web-страницы располагают интересной разновидностью меню – изображениями-картами, т.е. изображениями, чувствительными к нажатию кнопки мыши (imagemaps – от англ. слов image – изображение и map – карта, план).

Разные части изображения на странице могут быть связаны с разными URL-адресами. Так как пользователь должен знать, где расположены «чувствительные» области изображения, они часто выделяются рамками, тоже являющимися частью изображения.

Изображения-карты бывают двух типов: обслуживаемые сервером и программой-клиентом (браузером).

Синтаксис определения изображения-карты, обслуживаемой

клиентом, следующий:

<map name="mapname" >

<area [shape="shape"] coords="x,y,..." [href'="URL"] [nohref]>

</map>

Определение начинается тэгом <map> и заканчивается тэгом </map>. Для того чтобы ссылаться на это определение позже в тэге <img>, оно должно иметь имя, задаваемое при помощи атрибута name.

Для задания чувствительных областей используется тэг <area> табл.16.

Таблица 16. Атрибуты тэга <area>

Атрибут	Описание
shape	Определяет форму чувствительной зоны. Имеет значения rect, poly, circle, rect.
coords	В этом атрибуте перечисляются через запятую пары координат – x1,y1,x2,y2. Между парами также ставится запятая.
href	Определяет URL-адрес ссылки. Относительные адреса задаются относительно документа, содержащего тэг <map>. Если в этом же документе используется тэг <base>, адресация рассчитывается относительно URL, указанного в этом тэге.
nohref	Указывает нечувствительную зону, т.е. зону, не связанную с другими документами или ресурсами. Атрибуты nohref и href взаимоисключающие.

Ссылка на изображение-карту:

<map name=mymap>

<area shape=rect coords="0,0,100,100" href=item1.html>

<area shape=rect coords="101,0,200,100" href=item2.html>

<area shape=rect coords="201,0,300,100" href=item3.html>

</map>

<img src=mymap.gif usemap=#mymap>

Этот тэг загружает изображение mymap.gif. Атрибут usemap указывает на имя определения изображения-карты, которое было присвоено при помощи атрибута name тэга <map>.

Вставка объектов мультимедиа

Вставка видео

Для вставки видео фрагментов в код HTML-страницы и его дальнейшего просмотра средствами браузера используется уже известный Вам тэг <img> с новым атрибутом dynsrc. Например, строка кода: <img src=”image.gif” dynsrc=”video.avi”> выведет в окне браузера сначала картинку image.gif и весь текст, а затем, пока пользователь будет читать текст, догрузит видеофайл и запустит его вместо картинки.

При помощи атрибута loop можно задать количество раз воспроизведения клипа. Значение этого атрибута, равное –1 или символьной константе infinite, позволяет прокручивать данное изображение неограниченное колличество раз. Данный атрибут принимает целочисленные значения.

Атрибут start позволяет указать событие, после совершения которого начнется воспроизведение клипа. Например, <img src=”image.gif” dynsrc=”video.avi” start=mouseover/fileopen>

Для просмотра видеоклипа также можно использовать универсальный тэг <embed>. Он позволяет воспроизводить и видео- и аудио- клипы в обоих браузерах. Используя такие атрибуты данного тэга, как height и width, вы можете изменять высоту и ширину видеоклипа в пикселах. Используя ключевое слово autostart, можно автоматически загружать клип на выполнение сразу после его загрузки.

Атрибут loop=yes задает бесконечное количество воспроизведений данного клипа.

Вставка аудио

Один из наиболее популярных способов вставки аудиоклипов – использование простой ссылки на звуковой файл:

<img src=”images/sound.gif” align=left>

 Мое приветствие

В данном случаи загрузка и последующее воспроизведение данного аудиофайла будет происходить только после щелчка мышкой по ссылке. На экран этот файл выведен не будет.

Также используется тег <bgsound>. При использовании этого тега в вашем HTML-документе при его загрузке сначала происходит загрузка его текстовой и графической части и лишь затем – загрузка и воспроизведение аудиофайла указанного в атрибутах тега <bgsound>.

Например**:**

<bgsound=”myaudio.wav” loop=10>

Значение атрибута loop равное infinite, указывает на бесконечное количество повторений данного аудиофрагмента (пока пользователь не покинет страницу).

Для вставки звукого файла может быть использован тег <embed>. При выводе аудиоклипа броузер выводит на экран набор кнопок, позволяющий управлять воспроизведением клипа. Эти кнопки управления появятся на вашем экране только после того, как будет загружена вся HTML-страница и скачан целиком аудио файл.

<embed src=”lion.wav” height=60 width=140 autostart=true loop=true>

Данная строка выводит звуковой файл и устанавливает размер панели управления клипа. Атрибут loop позволяет задать количество раз проигрывания клипа.

Иногда еще используется атрибут hidden=true (по умолчанию – false), который, скрывает вывод на экран управляющик кнопок. В этом случае просто слышен и посетители не могут им управлять.

Таблицы стилей

Таблицы стилей HTML предназначены:

– для изменения расстояний междустроками, словами и отдельными символами;

– для установки левого, правого, верхнего и нижнего полей элемента (блока текста контейнера HTML);

– для установки отступа элемента;

– для изменения размера, стиля и других атрибутов шрифта элемента;

– для установки рамки вокруг элемента;

– для включения фонового изображения и фонового цвета в элемент.

Связывание документа с таблицей стилей

Большим преимуществом таблиц стилей является возможность отделить операцию форматирования от содержания документа. Сначала вы определяете, как должен выглядеть текст в том или ином месте страницы, а затем вводите сам текст. Если вы позднее решите, например, заменить цвет шрифта заголовков на синий, для этого будет достаточно поменять только стиль этих заголовков. Делать изменения в тексте нет необходимости табл.17.

Таблица 17. Способы применения таблицы стилей

Связывание (Linking).	Можно связать HTML-документ с таблицей стилей, хранящейся в отдельном файле.
Встраивание (Embedding).	Можно встроить таблицу стилей в HTML-документ с помощью контейнера <style>.
Задание стиля для отдельного фрагмента документа (Inline).	Можно определять элементы стиля «на лету», т. е. указывать их в тэгах HTML, например, в тэге абзаца <p>.

Создания таблицы стилей в виде отдельного файла для применения его ко всем страницам сайта. Этот метод упрощает создание сайта. Вы можете даже разработать единую таблицу стилей, которую могли бы использовать все сайты вашей организации.

Хранить таблицу стилей следует в текстовом файле с расширением .css. Его можно создать при помощи любого текстового редактора. У вас не будет никаких трудностей. Для связывания таблицы стилей с документом HTML используйте тэг <link> следующим образом:

<link rel=stylesheet href=www.myserver.com/mysheet.css type="text/css">

Укажите в атрибуте href URL-адрес вашей таблицы стилей. Дайте атрибуту type значение "text/css", что позволит программам просмотра, не поддерживающим таблицы стилей, проигнорировать указанный адрес.

Встраивание таблицы стилей в документ

Таблицу стилей необязательно хранить в виде отдельного файла. Ее можно встроить непосредственно в документ, однако в этом случае она будет действовать только внутри файла этого документа. Для распространения действия таблицы на другие документы ее необходимо скопировать в каждый из них.

Для включения таблицы стилей в документ воспользуйтесь контейнером <style>. Он размещается между тэгами <html> и <body>:

<head>

</head>

<style type="text/css"> Style definitions go here
</style>
<body>
</body>

Тэг <style> имеет единственный атрибут type, определяющий тип mime (Multipurpose Internet Mail Extension, стандарт электронной почты Internet). Определяйте его как "text/css" для того, чтобы браузеры, не поддерживающие таблицы стилей, могли игнорировать тэг <style>.

Ниже приведен конкретный пример тэга <style>.

<style type="text/css">
h1 {color: blue;}
</style>

Задание стиля для отдельного фрагмента документа:

Вы можете определять стиль, что называется, «на лету», оперативно внося требуемые изменения. Например, если вы определили стиль документа с заголовком одного цвета, а потом решили выделить цветом какой-то элемент заголовка, вы можете это сделать внутри тэга заголовка, не изменяя общий стиль документа.

Такой метод действует внутри тэга, где определен или переопределен стиль при помощи атрибута style. Он поддерживается всеми подчиненными тэгами тега <body>. Для оперативного определения стиля добавьте к нужному тэгу атрибут style и присвойте ему строковое значение, указывающее новый стиль: <h1 style="color: blue">

Используя атрибут style с тэгом , можно определять стиль части документа, расположенной в контейнере . Это работает благодаря принципу наследования. Например, если вы хотите установить цвет шрифта для целого блока тэгов синим, вы можете расположить эти тэги внутри контейнера и определить цвет шрифта текста следующим образом:

<h1>This is a heading</h1>
This is a paragraph. It will look blue in the user's browser

Для изменения стиля нескольких слов или даже символов можно использовать атрибут style совместно с тэгом , например:

This is a simple block of text

Синтаксис таблиц стилей

Таблицы стилей хранятся в текстовых файлах, удобных для редактирования. Их нетрудно создавать вручную, однако, как и для HTML-документов, существуют специальные редакторы таблиц стилей.

Таблицы стилей позволяют определять стиль для одного или нескольких тэгов. Например, вы можете создать таблицу стилей, определяющую стили для тэгов <h1>, <h2>, и . Каждое определение называется правилом (rule). Правило содержит селектор (тэг HTML), за которым следует декларация (определение стиля). Селектор является связующим звеном между определением и тэгом. Ниже приведен пример правила, указывающего стиль для каждого из тэгов заголовка

<h1>:

hl {color: blue;}

Декларация заключается в фигурные скобки. Каждая декларация имеет две части: название свойства и присваиваемое ему значение, разделенные двоеточием. В HTML включены десятки свойств (font-size, font-style, color, margin-right и т. д.). Каждое свойство может принимать несколько значений, одно из которых приписывается ему по умолчанию.

В предыдущем примере было указано лишь одно свойство color. Однако ничто не мешает определить целый ряд свойств в одном тэге, отделив их друг от друга точкой с запятой:

Hl {color: blue; font-size: 12pt; text-line: center;}

В этом примере программа просмотра выведет каждый заголовок первого уровня синим шрифтом размером 12 пунктов и выровняет их по центру окна. Для всех прочих свойств будут использоваться значения по умолчанию (например, свойству font-style будет присвоено значение normal).

Группирование селекторов

Если вы хотите определить один и тот же стиль для нескольких тэгов, вы можете перечислить их в отдельном списке:

p {font-size: 12pt;}
ul {font-size: 12pt;}
li {font-size: 12pt;}

HTML позволяет сделать то же самое и в более компактном виде – в одной строке: p, ul, li {font-size: 12pt;}. Запятая здесь является обязательным элементом. Если она опущена, смысл правила изменится.

Комментирование таблицы стилей

По мере усложнения таблицы стилей, скорее всего, понадобится включить в нее дополнительные сведения о назначении того или иного правила. Комментарии располагаются между символами **/*** и ***/** и игнорируются программами просмотра, например:

body {margin-left: lin;} /* Отступ на 1 дюйм */
h1 {margin-left: -lin;} /* Сдвиг влево на 1 дюйм */
h2 {margin-left: -lin;} /* Сдвиг влево на 1 дюйм */

Комментарии могут иметь вид блоков текста, дающих подробное описание стиля страницы, например: /*------.

Свойство margin-left тэга <body> установлено в 1 дюйм. Так как все внутренние тэги наследуют это значение, то вся страница будет иметь отступ в 1 дюйм. Заголовки первого и второго уровней имеют отрицательный отступ (-1 дюйм), т.е. будут прижаты к левой границе окна браузера.

-----*/
body {margin-left: lin;} /* Отступ на 1 дюйм */
h1 {margin-left: -lin;} /* Сдвиг влево: на 1 дюйм */
h2 {margin-left: -lin;} /* Сдвиг влево на 1 дюйм */

Наследование свойств

В HTML подчиненные тэги наследуют некоторые свойства родительских тэгов. Например, все тэги контейнера <body> (<p> и <ul>) будут обладать некоторыми свойствами тэга <body>. Точно так же тэг <li> наследует свойства тэга <ul>. Рассмотрим следующий код:

```
<style type="text/css"> p{color: blue;}
</style>
<body>
<p>Hello. This is a paragraph of text. <em>This is emphasized</em><p>
</body>
```

Таблица стилей этого документа устанавливает цвет в тэге <p> синим, однако, цвет для тэга явно не определен (по умолчанию – это черный цвет). Здесь не о чем беспокоиться, так как этот тэг находится в родительском контейнере <p> и наследует таким образом синий цвет.

Применение контекстных селекторов

Иногда возникает необходимость определения двух (и более) стилей для одного тэга. Например, может понадобиться указание двух стилей для тэга <li>: один для случая, когда он подчинен тэгу <ul>, и второй – когда он подчинен тэгу <ol>. Это возможно сделать с помощью контекстных селекторов. Контекстный селектор определяет точную последовательность тэгов, для которых будет применен тот или иной стиль. Другими словами, вы можете указать, что какой-то стиль должен применяться, например, в тэге <li> только в том случае, если этот тэг является подчиненным тэгу <ol>: ol li{list-style-type: decimal;}.

Для того же тэга <li> можно определить другой стиль, действительный только в случае подчиненности тэгу <ul>: ul li {list-style-type: square;}.

Заметьте, что список селекторов не разделен запятыми. В противном случае всем тэгам списка будет приписан один и тот же стиль.

Почему таблицы стилей HTML называются каскадными

В рекомендациях W3C таблицы стилей называются «каскадными таблицами стилей» потому, что для верстки Web-страницы можно применять не одну, а сразу несколько таблиц. При этом программа просмотра сама определяет последовательность использования таблиц и разрешает конфликты между ними по принципу каскадирования.

Как это работает? Каждому правилу браузер приписывает весовой коэффициент. При интерпретации каждого тэга программа просматривает все правила этого тэга и сортирует их по величине весового коэффициента. Выигрывает самое «весомое» правило.

Существуют следующие общие принципы разрешения конфликтов между таблицами стилей:

Таблица стилей автора страницы «весомее» таблицы стилей читателя, которая, в свою очередь, «весомее» установок браузера по умолчанию.

Старшинство типов таблиц стилей в документе (по убыванию): текущее задание стиля (inline), встраивание (embedding) связывание (linking).

Использование классов в таблицах стилей

Классом называется определение нескольких стилей одного элемента, каждый из которых может использоваться в нужном месте страницы. Например, вы можете определить три вариации стиля заголовка h1. Определение вариаций похоже на указание стиля, только к названию тэга добавляется произвольное имя класса, отделенное точкой:

h1.blue {color: blue;} h1.red {(color: red;} h1.black {color: black;}.

Теперь, включая в документ тэг <h1>, можно указать в нём конкретный стиль при помощи атрибута class: <h1 class=red>Red Heading</h1>.

Вы можете разрешить обратиться к какому-либо классу из любого тэга, если при определении данного класса опустить в селекторе имя тэга, например: red {color: red;}.

Блочная верстка страниц

Отличия блочной верстки от табличной

Верстка блоками, с помошью такого тега как div, имеет ряд больших преимуществ по сравнению с версткой таблиц (с помощью элемента table), среди них следующие:

1. Дизайн сверстаный блоками быстрее загружается.
2. Содержимое блоков в отличие от содержимого ячеек таблиц отображается по мере загрузки(содержимое таблиц же, напротив, отображается только тогда, когда загрузится все содержимое таблицы).
3. Код, написаный блоками, имеет более читаемый вид.

Управление положением блоков на странице

Для решения этого вопроса используется такое свойство как float.

Свойство float может принимать три значения:

1. left - выравнивание элемента по левому краю страницы;
2. right - выравнивание элемента по правому краю страницы;
3. none - элемент страницы ни куда не перемещается, то есть будет там где он должен быть. Это значение используется по умолчани.

Свойство clear может принимать четыре значения:

1. left - установка элемента ниже любого предыдущего, перемещенного влево блока;
2. right - установка элемента ниже любого предыдущего, перемещенного вправо блока;
3. both - установка элемента ниже любого предыдущего перемещенного блока;
4. none - нет ни каких ограничений на положение блока относительно перемещаемых блоков.

Формы

С помощью HTML вы можете создавать простые формы, предполагающие ответы типа «да» и «нет», вы можете разрабатывать сложные формы для заказов или для того, чтобы получить от своих читателей какие-либо комментарии и пожелания.

Форма представляет собой несколько полей, где пользователь может ввести некоторую информацию либо выбрать какую-то опцию. После того

как пользователь отправит информацию, она обрабатывается программой (скриптом), размещённой на сервере. Скрипт – это короткая программа, специально созданная для обработки каждой формы.

Формы HTML позволят вам получать информацию от читателей.

В форму могут заноситься мнения сторон дискуссионной группы.

Работа с тэгами форм

В HTML существует три тэга для создания различного типа полей в форме: <textarea>, <select> и <input>. Любое их количество может быть размещено в контейнере между тэгами (табл.18) <form> и </form>.

Таблица 18. Тэги форм

<textarea>	Определяет поле, в которое пользователь вводит многострочную текстовую информацию.
<select>	Позволяет пользователю сделать выбор в окне с полосой прокрутки либо в раскрывающемся меню.
<input>	Обеспечивает некоторые другие виды ввода информации: ввод одной строки текста, установку и сброс флажков (checkboxes), выбор переключателя (radio buttons) и нажатие кнопки для отправки данных или очистки формы.

Каждая форма начинается тэгом <form>. В нем нужно определить два атрибута, указывающих используемый скрипт и метод посылки данных:

action	Определяет URL, который примет и обработает данные формы. Если этот атрибут не определен, данные отправляются но адресу страницы, на которой помещена форма.
method	Указывает форме, как послать информацию соответствующей программе обработки (скрипту). Обычно он получает значение post, тогда информация формы посылается отдельно от URL. Если указано значение get, информация формы посылается вместе с URL. Get для посылки данных до 256 символов, а post для посылки данных длинной более 256 символов.

Например:

```
<form method="post" action="/cgi-bin/comment script">
...
</form>
```

В этом примере дано указание браузеру отправить заполненную форму для обработки скриптом comment script, расположенным в каталоге cgi-bin вашего сервера, и использовать метод посылки post.

Метод get – данные, посылаемые браузером серверу, включают модифицированный URL- адрес: метод(http), servr:port и в конец добавляется символ ?, далее следует строка запроса.

Метод post – запрос серверу посылается как mime-данные, при этом пробелы заменяются символом +, поля разделяются & и символы кодируются в шестнадцатиричном коде в виде %xx.

На странице можно расположить любое число форм, однако нужно следить за тем, чтобы не поместить одну форму в другую.

Тэг <textarea> предназначен для построения поля для ввода многострочной текстовой информации. При помощи атрибутов rows и cols этого тэга можно построить поле любого размера. В контейнере textarea допускается размещать любой текст, который будет выведен в поле ввода по умолчанию.

Поле textarea удобно тем, что пользователь может ввести в него любое количество информации (табл.19).

Таблица 19. Атрибуты тэга <textarea>

name	Обязательный атрибут, определяющий название информации.
rows	Устанавливает высоту поля, т.е. число строк в нём.
cols	Устанавливает ширину поля, т.е. длину строки.

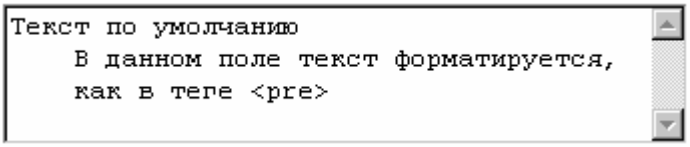
Хотя атрибуты rows и cols не являются обязательными, они не имеют определенных значений по умолчанию (для каждого браузера эти значения различны), поэтому лучше их всегда указывать.

Данный тэг имеет еще один атрибут – wrep. Значения, которые он может принимать, приведены в таблице 20.

Таблица 20. Значения атрибута wrep

off	Переход на новую строку только при нажатии на Enter. Данные отправляются одной строкой.
soft	Автоматический перенос текста на новую строку, при достижении границы текстового поля. При этом текст на сервер передается одной строкой. Это значение по умолчанию.
hard	Визуально так же как soft, но на сервер будет отправленно несколько строк.

```
<form>
<textarea name="comments" rows=4 cols=40>
Текст по умолчанию
В данном поле текст форматируется,  как в теге <pre>
</textarea>
</form>
```



Тэг <select> используется для создания всплывающего меню или списка опций с полосой прокрутки. Список опций и пункты меню располагаются внутри контейнера select. Как и тэг <textarea>, тэг <select> требует обязательного определения имени в атрибуте name. Количество опций указывается в атрибуте size. В табл.21 перечислены атрибуты тэга <select>.

Таблица 21. Атрибуты тэга <select>

name	Определяет имя поля
size	Определяет вертикальный размер окна для опций выбора. Если атрибут опущен или его значение равно 1, выводится всплывающий список опций(раскрывающийся список). Если указано число больше единицы, то опции выводятся в окне с полосой прокрутки. Если значение атрибута больше, чем фактическое количество элементов списка, добавляются пустые строки. При их выборе пользователем возвращаются пустые поля.
multiple	Этот атрибут позволяет производить выбор сразу нескольких опций.

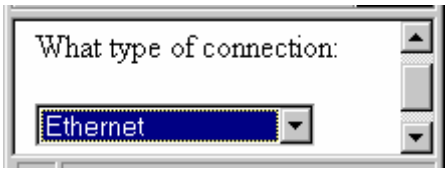
Список опций включается в контейнер <select> при помощи тэгов

<option>. Этот тэг имеет два атрибута (табл.22).

Таблица 22. Атрибут тэга <option>

value	Указывает значение, возвращаемое программе обработки (скрипту) в случае выбора опции пользователем.
selected	Указывает на опцию, выделенную по умолчанию.

```
<form>
<select name="network">
<option selected value="ethernet"> Ethernet
<option value="token16"> Token Ring - 16MB
<option value="token4"> Token Ring - 4MB
<option value="localtalk"> LocalTalk
</select>
</form>
```



Тэг <select> можно использовать как дополнительное средство навигации. Для этого в него нужно включить список URL-адресов. После выбора одного из адресов и нажатия кнопки Отправить (Submit), скрипт, размещенный на вашем сервере или на машине читателя, загрузит требуемую страницу.

Тэг <input> в отличие от <textarea> и <select> является одиночным тэгом. Он предназначен для сбора информации различными способами, включая текстовые поля, поля для ввода пароля, переключатели, флажки, кнопки для отправки данных (Submit) и для очистки формы (Reset, Clear).

Тэг <input> располагает следующими атрибутами (табл.23).

Таблица 23 – Атрибуты тэга <input>

namesize	Указывает размер поля ввода в символах.
maxlenght	Определяет максимально возможное число вводимых в поле.
value	Для текстового поля определяет текст, выводимый по умолчанию. Для флажко переключателей указывает значение, возвращаемое программе обработки. Для кно отправки и очистки формы определяет надпись на кнопке.
checked	Устанавливает флажок или переключатель во включенное состояние по умолчанию другими типами тэгов <input> не употребляется.
type	Устанавливает тип поля ввода.

Атрибут type тэга <input> может принимать следующие значения:

1. text.

Является значением по умолчанию и предполагает создание одной строки для ввода данных. Для этого типа поля ввода употребляются атрибуты name (обязательный), size, maxlenght и value.

```
<form>
Please specify:<input type="text" name="network_other">
</form>
```

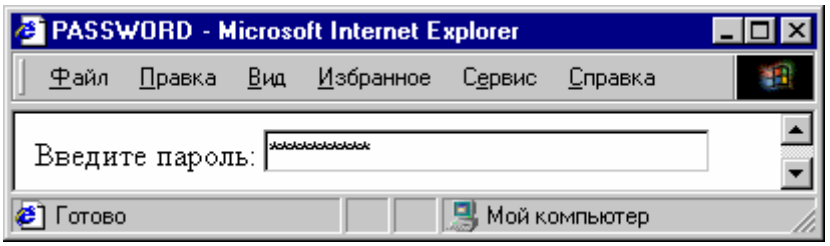
2. password.

Позволяет заменять вводимые символы пароля звездочками. Для этого типа поля ввода используется атрибуты name (обязательный), size, maxlenght и value.

<form>

Введите пароль: <input type="password" name="secret_word" size="10" maxlenght="30">

</form>



3. checkbox.

Позволяет вывести поле для установки флажка в виде маленького квадратика, в котором может быть произведена отметка опции «галочкой». Может использоваться совместно с атрибутами name (обязательный), value и checked (определяет установленный по умолчанию флажок). Флажки обычно употребляются, когда можно выбрать сразу несколько опций из числа предложенных.

<form>

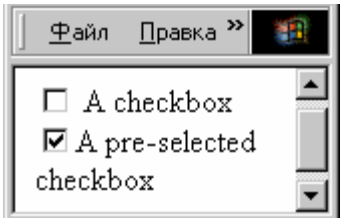
<input type="checkbox" name="checkbox1" value="checkbox_value1">

A checkbox

<input type="checkbox" name="checkbox2" value="checkbox_value2" checked>

A pre-selected checkbox

</form>



4. radio.

Позволяет выбрать только одну из представленного числа опций. Переключатели можно группировать, задавая одно и то же значение атрибута name (обязательный). Также используются атрибуты value и checked. Значение атрибута value отправляется на сервер для обработки скриптом.

Form #1:

<form>

<input type="radio" name="choice" value="choice1"> Yes.

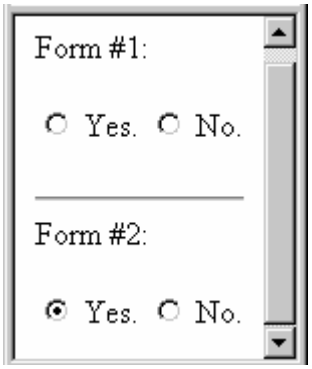
<input type="radio" name="choice" value="choice2"> No.

</form>

<hr>

Form #2:

```
<form>
<input type="radio" name="choice" value="choice1" checked> Yes.
<input type="radio" name="choice" value="choice2"> No.
</form>
```

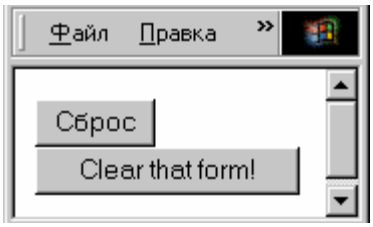


Если опций для выбора слишком много, для экономии места лучше использовать тэг <select>.

5. reset.

Позволяет создать кнопку для очистки формы. Атрибут value может быть использован здесь для наименования этой кнопки (по умолчанию кнопка имеет надпись “Сброс” или “Reset”).

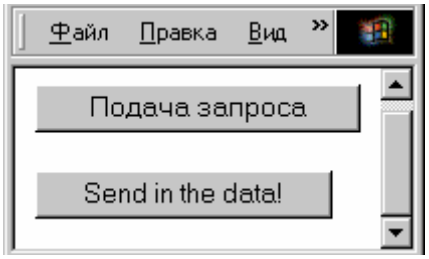
```
<form>
<input type="reset">
<br>
<input type="reset" value
="Clear that form!">
```



6. submit.

Используется для создания кнопки, по нажатию которой введенные данные отправляются на сервер для обработки программой-скриптом. В атрибуте value может быть указано название для этой кнопки (по умолчанию – “Submit” или “Подача запроса”).

```
<form>
<input type="submit">
<br>
<input type="submit" value="Send in the data!">
</form>
```



7. button.

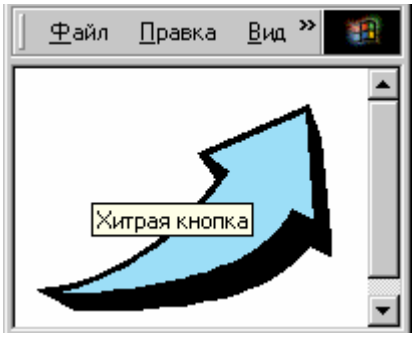
Используется для создания кнопки при нажатии на которую будет выполнятся отдельный скрипт. В качестве параметров можно задать размеры кнопки и ее подпись.

```
<form>
<input  type="button" value="TEST" width="200" height="100">
</form>
```

8. image.

Используется для создания кнопки в виде картинки, при щелчке левой кнопкой мыши по которой все данные из формы будут отправлены на сервер. Таким образом, это подобие кнопки submit, только оформленной по-другому.

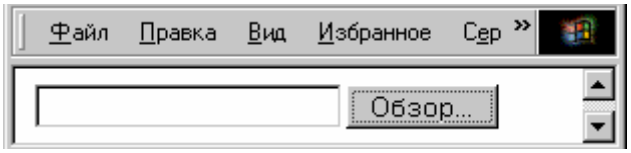
```
<form>
<input type="image" src="trigger.gif" align="right"  alt="Хитрая кнопка">
</form>
```



9. file.

Используется для обращения к файловой структуре диска и выбора файла, который необходимо переслать по компьютерной сети. Пересылка осуществляется в двоичном коде.

```
<form>
<input type="file">
</form>
```



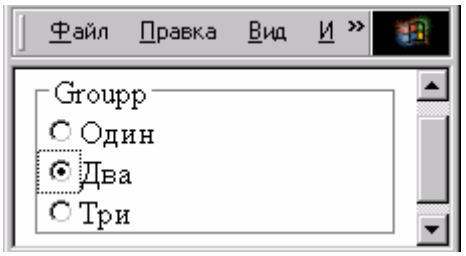
10. hidden.

Используется для передачи данных на сервер из многоступенчатых форм без участия пользователя. Данные в этих полях на экране не отображаются.

Выделение нескольких элементов в группу

Парный тэг <fieldset> предназначен для выделения нескольких элементов управления в группу. В паре с этим тэгом работает парный тэг <legent>, при помощи которого задается подпись для этой группы. У тэга <legent> есть атрибут align, который может принимать значения left, right, center и предназначеный для выравнивания подписи группы относительно экрана.

```
<fieldset>
<legend align=left>Groupp</legend>
<input type="radio" name="radio1" value="1">Один<br>
<input type="radio" name="radio1" value="2">Два<br>
<input type="radio" name="radio1" value="3">Три<br>
</fieldset>
```



Размещение в документе форм нескольких типов

Расположение на странице форм разных типов способно сделать ее более выразительной и понятной.

Размещение на странице нескольких форм требует их визуального разделения. Для этого можно использовать тэг <hr>, создающий горизонтальные линии, или тэг <img> для включения узкого горизонтально расположенного изображения

2.3. Основы языка XHTML

Как и HTML, XHTML является подмножеством языка SGML (*Standard Generalized Markup Language*, стандартный обобщённый язык разметки) - метаязыка, на котором можно определять язык разметки для документов (рис. 6). От SGML произошли HTML и XML (*Extensible Markup Language*, расширяемый язык разметки). HTML – это конкретное приложение SGML, а XML – это подмножество SGML, разработанное для упрощения процесса машинного анализа документов.

HTML не вполне совместим с XML, поскольку менее строг к синтаксису, и не может считаться его подмножеством. Расширяемый язык разметки гипертекста XHTML в отличие от обычного HTML является подмножеством языка XML. Фактически, XHTML - это HTML, записанный в соответствии с синтаксическими правилами языка XML.

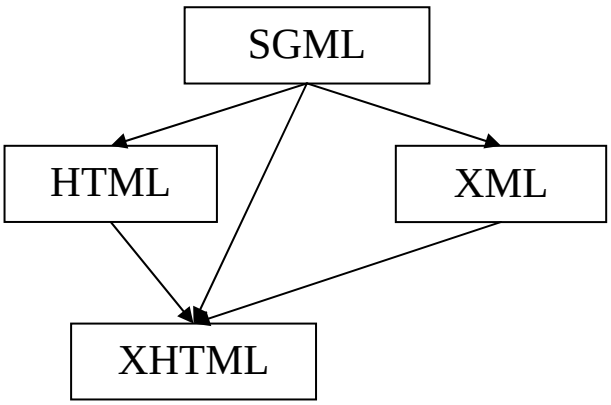


Рис.6. Эволюция языков разметки

XML – это обобщенный язык разметки, он не имеет набора заранее определенных тегов. В отличие от HTML XML позволяет создавать собственные теги и таким образом формировать собственные структуры. XML – текстовый формат, предназначенный для хранения произвольных структурированных данных (взамен существующих форматов файлов баз данных), а также для организации универсального обмена данными между приложениями.

Документ XML – это описанная в текстовом формате иерархическая структура, предназначенная для хранения произвольных данных. Визуально структура может быть представлена как дерево элементов. Элементы XML описываются тегами.

Приведем в качестве примера возможное библиографическое описание книги с помощью тегов.

```
<?xml version="1.0" encoding="UTF-8"?>
<book id="1" date="12/04/99">
<title>Собрание сочинений</title>
<author>Стругацкий Аркадий</author>
<author>Стругацкий Борис</author>
<publisher></publisher>
<year>1996</year>
</book>
```

Другой пример – описание простого кулинарного рецепта, размеченное с помощью XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<recipe name="хлеб" preptime="5" cooktime="180">
<title>Простой хлеб</title>
<ingredient amount="3" unit="стакан">Мука</ingredient>
<ingredient amount="0.25" unit="грамм">Дрожжи </ingredient>
<ingredient amount="1.5" unit="стакан">Теплая вода </ingredient>
<ingredient amount="1" unit="чайная ложка">Соль </ingredient>
<instructions>
<step>Смешать все ингредиенты и тщательно замесить. </step>
<step>Закрыть тканью и оставить на один час в теплом помещении.</step>
<step>Замесить еще раз, положить на противень и поставить в духовку.</step>
    </instructions>
```

</recipe>

Чтобы утвердить семантические правила нового языка, то есть список допустимых элементов структуры данных, их возможное содержимое и атрибуты, необходимо создать DTD-определения (DTD-схему). Например, можно описать, что внутри структуры book может быть единственный элемент title и year, а элементов author и publisher может быть несколько. Кроме того, можно указать, является ли элемент обязательным, допускает ли он пустое содержимое. Можно также задать порядок следования элементов в структуре.

DTD (*Document Type Definition*, определение типа документа) – описывает схему документа для конкретного языка разметки с точки зрения семантических ограничений этого документа. Также DTD может объявлять конструкции, которые всегда необходимы для определения структуры документа.

Документ XML, написанный в соответствии со всеми общими правилам синтаксиса XML, считается *правильно построенным* (well-formed). Документ, который неправильно построен, не может считаться документом XML; XML-процессор (*парсер*) не должен обрабатывать его обычным образом и обязан классифицировать ситуацию как «фатальная ошибка».

Следующий уровень правильности документа XML – *действительный* (валидный, valid) документ. Действительный документ дополнительно соответствует некоторым семантическим правилам. Это более строгая дополнительная проверка корректности документа на соответствие заранее определённым, но уже внешним правилам, например, структуры и состава данного, конкретного документа или семейства документов. Именно эти правила и задаются DTD-схемой.

В противоположность XML HTML гораздо более строго определенный язык разметки с ограниченным набором тегов. В любом случае, общий характер XML позволяет рассматривать HTML-документы как XML-документы с набором тегов для отображения в Web-браузерах. Однако старые стандарты HTML не до конца совместимы с XML. Например, в HTML необязательно закрывать тег p, то есть закрывающую часть тега </p> можно опускать. Web-браузер сможет правильно интерпретировать эту конструкцию, поскольку он так запрограммирован, однако XML-парсер выдаст ошибку о том, что HTML-документ не является правильно построенным (well-formed).

Чтобы устранить разрыв между этими двумя языками разметки, и был разработан XHTML. По существу это обычный HTML, в который добавили синтаксические правила XML для создания well-formed документов.

Структура XHTML-документа

Корневым элементом документа должен быть html.

Корневой элемент документа должен обозначить пространство имён XHTML путём использования атрибута xmlns (XMLNAMES). Пространство имён XHTML определено в http://www.w3.org/1999/xhtml.

В документе должно присутствовать объявление doctype, предшествующее корневому элементу html. Публичный идентификатор,

включённый в объявление doctype, обязан быть ссылкой на одно из определений типа документа.

Каждый документ разделен на две части: *головную* часть (head) и основную часть или *тело* (body) документа (рис. 7).

В документах с фреймами (тип документа Frameset) тело body может отсутствовать (заменено фреймовой структурой).

Головная часть содержит служебную информацию о документе, предназначенную для браузера или поисковых систем, и не отображается на экране.

Все содержимое Web-страницы, предназначенное для вывода на экран и просмотра пользователем, помещается в тело документа body. Для того чтобы отделить эти части друг от друга, используются соответствующие теги.

Тег title является обязательным и указывается в разделе head. Элемент title задает название (заголовок) Web-страницы. Это название отображается в строке заголовка (в самом верху) окна браузера. Кроме того, содержимое title используется как название Web-страницы при ее внесении пользователем в список «Избранное» («Закладки»), а также в результатах поиска, выдаваемых поисковыми системами. Поскольку пользователи часто обращаются к документам вне контекста, следует задавать осмысленные заголовки.

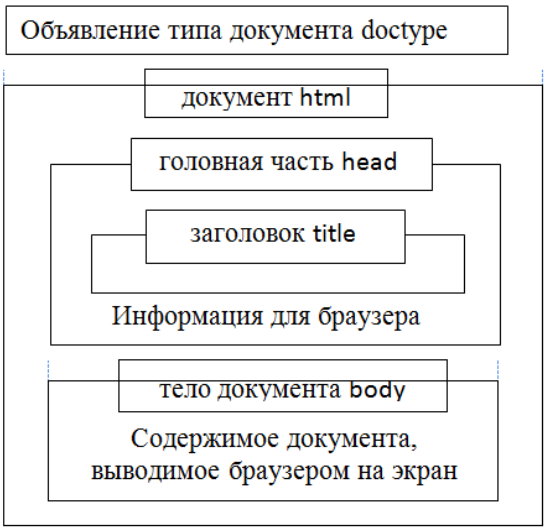


Рис. 7. Структура XHTML-документа

Таким образом, минимальный XHML-документ может выглядеть следующим образом:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=windows-1251">
<title>Заголовок документа</title>
</head>
<body>
```

Текст, выводимый на экран
</body>
</html>

В данном примере головная часть документа содержит необязательный тег meta, задающий кодировку Web-страницы (для отображения русскоязычных символов). При просмотре Web-страницы в окне браузера будет выведена всего одна строка: Текст, выводимый на экран.

Форматирование текста

Теги форматирования могут разбивать текст на блоки (абзацы, списки, таблицы) или определять написание символов текста внутри блока.

Поэтому все теги делятся на теги уровня блока и теги уровня текста. С точки зрения корректности написания XHTML-кода важно следить, чтобы теги уровня текста обязательно указывались внутри какого-либо блока.

Основные теги XHTML, используемые для форматирования абзацев, приведены в табл. 24.

Таблица 24. Основные теги форматирования абзацев в XHTML

Теги	Описание и особенности использования
p	Абзац, отделенный от остального текста пустыми строками. p является тегом уровня блока, однако не может содержать элементы уровня блока (включая сам p). Не рекомендуется использовать пустые элементы p
div	Абзац, не имеющий никакого дополнительного форматирования. Обычно используется для группировки блоков
h1 - h6	Шесть уровней заголовков: от h1 (самый верхний) до h6 (самый нижний). Визуально браузеры обычно отображают более значительный заголовок более крупным шрифтом. Элементы заголовков являются частью глобальной структуры документа и предназначены для краткого описания смысла последующего раздела Теги заголовков могут считываться автоматическими средствами представления документа для создания автоматического оглавления. Важно соблюдать правильный порядок следования уровней заголовков для описания разделов и подразделов с целью их корректной обработки при автоматическом считывании
blockquote	Блок с дополнительным отступом. Выделяется, как и p, пустыми строками
pre	Преформатированный текст - блок, сохраняющий форматирование текста в коде: разбиение на строки, табуляции и пробелы (может отображаться браузером моноширинным шрифтом). Внутри блока pre не действует ряд тегов (img, object, big, small, sub или sup), тег p действует как br.
br	Непарный тег, принудительно обрывает (оканчивает) текущую строку текста.
hr	Непарный тег, вставка горизонтальной линии.

Тег или параметр, задающий отступ первой строки абзаца (красную строку), не определен. Красная строка может быть задана с помощью таблиц стилей.

Ширина блоков автоматически подстраивается под размер окна браузера (также как абзацы Word подстраиваются под ширину документа за счет автоматического переноса на новую строку). Перенос текста осуществляется по словам. Дополнительную возможность переноса можно задать, используя символ «мягкого дефиса» (или).

В случае если, напротив, необходимо запретить разрыв строки в каком-то месте, можно использовать символ неразрывного пробела   (символы-мнемоники   или).

Теги уровня текста используются для задания специфического написания последовательности символов (указание текста в кавычках, верхние/нижние индексы) или задания параметров шрифта (выделение жирным, курсивом, крупный, мелкий шрифт). Вместе с тем для форматирования шрифтов предпочтительным считается использование таблиц стилей.

Основные теги форматирования символов представлены в табл. 25

Таблица 25. Основные теги форматирования символов

Теги	Описание	Теги	Описание
q	Текст, заключенный в кавычки	big	Крупный шрифт
tt	Моноширинный шрифт	small	Мелкий шрифт
b	Жирный шрифт (*bold*)	ups	Верхний индекс
i	Курсив (*italic*)	sub	Нижний индекс

Рис.8. Форматирование символов тегами XHTML

Фрагмент кода «Форматирование символов тегами XHTML»:

```
<p>Обычный текст, <q>текст в кавычках</q>,
<tt>моноширинный</tt>, <b>жирный</b>, <i>курсив</i>,
<b><i>жирный курсив</i></b>, <big>крупный</big>,
<small>мелкий</small>,
верхний<sup>индекс</sup>,
нижний<sub>индекс</sub></p>.
```

2.4. Каскадные таблицы стилей. Синтаксис CSS

Таблицы стилей (или каскадные таблицы стилей, CSS) – это описание правил, задающих параметры представления отдельных элементов на языке HTML

CSS появились одновременно с HTML 4.0 (Dynamic HTML). Сам термин «каскадные таблицы стилей» был предложен в 1994 году.

Стиль оформления – это все то, что определяет внешний вид документа: размер, цвет и вид шрифта текста, цвет фона текста, наличие границ, подчеркивания, выравнивание текста и т.д. Стиль определяется набором правил отображения тегов, задаваемых таблицей стилей.

Таблица стилей – это шаблон, который управляет форматированием тегов HTML в Web-документе. Таблица стилей состоит из набора *правил* описания стиля.

Любое *правило* каскадных таблиц стилей имеет две части: *селектор* и *определение*. *Селектор* – это любой элемент или группа элементов web-

страницы, для которых определяется форматирование. *Определение* описывает конкретный вид форматирования и состоит из двух частей: *свойства* и *значения*, разделенных знаком двоеточия (рис. 9).

Например, цвет текста абзаца может быть задан с помощью стиля как: Текст

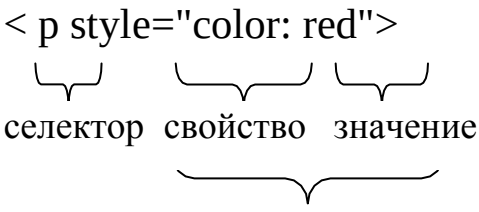


Рис.9. Правило таблицы стилей

В ряде случаев использование CSS позволяет создать более компактный и эффективный код HTML-документа.

В одном правиле можно задать несколько определений, разделенных знаком точки с запятой (;). Например, задание красного цвета, жирного написания и выравнивания «по ширине» текста абзаца с помощью стиля выглядит так:

 Текст

Некоторые свойства стиля приведены в таблице 26.

Таблица 26. Свойства стиля

Свойство	**Описание свойства**	**Значения**
font-family	Тип шрифта	Имя шрифта, список имен
font-size	Размер шрифта	Относительные размеры (small, large, x-large и др.), %, pt, px, mm, cm, in, em и т. п.
font-weight	Толщина шрифта	normal, bold и т. п.
font-style	Наклон шрифта	normal, italic и т. п.
text-transform	Изменение регистра	lowercase, uppercase, capitalize, none
text-decoration	Подчеркивание	underline, overline, line-through, none
color	Цвет текста	цвет
background-color	Цвет фона	цвет
background-image	Фоновый рисунок	url(«URL рисунка»)
background-repeat	Повторение фонового рисунка (мозаика)	no-repeat, repeat-x, repeat-y, repeat
background-position	Позиционирование фоновой картинки	left, right, center, top, bottom
text-indent	Красная строка	%, pt, px, mm, cm, in, em и т. п.
Свойства стиля border-style	Тип рамки	double, solid, dashed, dotted, none, outset, inset, ridge
border-width	Толщина рамки	thin, medium, thick, %, pt, px, mm, cm, in, em и т. п.
border-color	Цвет рамки	цвет
word-spacing	Разрядка слов	normal, %, pt, px, mm, cm, in, em и т. п.
letter-spacing	Разрядка символов	normal, %, pt, px, mm, cm, in, em и т. п.

white-space	Управление переносом на новую строку	normal, nowrap, pre, pre-line, pre-wrap
vertical-align	Выравнивание по	top, bottom, baseline, middle
text-align	Выравнивание текста	left, right, center, justify
padding	Отступы (внутренние)	%, pt, px, mm, cm, in, em и т. п.
margin	Границы блока (внешние отступы)	%, pt, px, mm, cm, in, em и т. п.
visibility	Видимость	visible, hidden
display	Отображение на экране	block, inline, none
overflow	Выход контента за границы элемента	hidden, scroll, visible
float	Обтекание текстом	left, right, none
clear	Запрет обтекания	left, right, none, both
position	Тип позиционирования	absolute, fixed, relative, static
z-index	Позиция по «глубине»	неотрицательное целое число
top, right, bottom, left	Позиция элемента	%, pt, px, mm, cm, in, em и т. п.
width, height	Размеры элемента	%, pt, px, mm, cm, in, em и т. п.

Большинство свойств стиля поддерживает также значение inherit-наследование от родительского элемента (контейнера).

Возможности форматирования, предоставляемые CSS, значительно превосходят средства языка HTML как такового; кроме того, использование таблиц каскадных стилей в сочетании со сценариями позволяет добиваться различных динамических эффектов, то есть форматирование страницы может изменяться со временем или в результате каких-то пользовательских действий.

Описание стиля может быть вынесено за рамки конкретного экземпляра тега, то есть задано для всех экземпляров тега или для группы элементов. В этом случае изменение форматирования группы элементов производится однократным внесением изменений в общее описание стиля.

В целом, механизм таблиц каскадных стилей позволяет:

– использовать дополнительные возможности форматирования;

– сократить число используемых в документе тегов;

– задавать более гибкое форматирование, удобное для модификации. Существует несколько версий технологии CSS: оригинальная версия - CSS1, улучшенная версия - CSS2, исправленная - CSS2.1, версия, разрабатываемая в настоящее время, CSS3.

Не все браузеры поддерживают технологию CSS. Современные версии основных браузеров (IE версии 8 и выше, Mozilla Firefox, Opera, Safari, Google Chrome) обеспечивают ее наиболее полную поддержку. Для проверки поддержки браузером Web-стандартов (в том числе и различных частей стандарта CSS) был разработан тест Acid, его версии: Acid2, Acid3 (http://www.acidtests.org/).

Указание правил CSS в HTML-документе может производиться четырьмя разными способами (табл. 27).

Таблица 27. Варианты использования стилей CSS

Область охвата	Тип таблицы стилей	Вариант использования
Отдельный экземпляр тега /отдельный элемент страницы	Стиль, встроенный в тег	Указание стиля внутри тега
Все или несколько экземпляров тега / группа элементов	Внутренняя таблица стилей	Внедрение стиля (описание стиля - в разделе head страницы)
Несколько страниц сайта	Внешняя таблица стилей	Связывание
Страницы нескольких сайтов		Импорт

Встраивание стиля в теги документа производится с помощью параметра style и позволяет изменить форматирование конкретных экземпляров тегов страницы.

Примеры встраивания стилей в тег p приведены выше.

Внедрение стиля позволяет определять форматирование для всех экземпляров тега или для группы элементов страницы. Таблица стилей внедряется в текст HTML-документа в раздел заголовка head с помощью тега style.

Пример:

```
<html>
<head>
<title>Внедренная таблица стилей</title>
<style type="text/css"> p {       color : red;
font-weight : bold; text-align:justify
}
</style>
</head>
<body>
<p>Общий стиль всех абзацев</p>
</body>
</html>
```

Связывание с таблицей стилей: таблица стилей сохраняется в виде внешнего файла, а в HTML-документ помещается ссылка на нее. Ссылка на файл с таблицей осуществляется тегом link, который указывается в разделе заголовка head. Связывание позволяет использовать одну таблицу CSS для форматирования элементов сразу нескольких Web-страниц.

Пример связывания с файлом стилей.

Содержание файла таблицы стилей (файл с расширением CSS, например, st.css):

```
p {       color : red;
font-weight : bold; text-align:justify
}
```

Указание в коде Web-страницы ссылки на файл таблицы стилей st.css, расположенной в корневой папке сайта:

```
<html>
<head>
<title>Связанная таблица стилей</title>
<link rel="stylesheet" type="text/css" href="/st.css">
```

```
</head>
<body>
<p>Текст абзаца</p>
</body>
</html>
```

Импорт таблицы стилей – указание месторасположения в сети (URL) файла с таблицей CSS. Вместо тега link в разделе head указывается тег import формата @import url(«URL-адрес»); например:

```
@import url('/st.css');
```

Таблицы стилей называются каскадными из-за порядка применения различных стилей к элементам страницы. Каскад распространяется сверху вниз: сначала внешняя таблица стилей, потом внутренняя, потом стиль, встроенный в тег.

Чем ближе правило к элементу, тем выше его приоритет. Приоритет стиля важен, когда для элемента заданы несколько противоречащих друг другу правил. Приоритет стилей (в порядке убывания) следующий:

1. Стиль встроенный в тег.
2. Внедренная в страницу таблица стилей.
3. Внешняя таблица стилей (связанная, импортируемая).
4. Стандартные настройки браузера.

На практике разработчики отдают предпочтение какому-либо одному типу таблиц стилей, обычно внешним таблицам, связанным со страницей.

Верстка

Наиболее популярным является деление макетов по ширине и количеству колонок. Выделяют следующие типы макетов, связанных с шириной:

- фиксированные;
- резиновые;
- эластичные;
- адаптивные;
- комбинированные.

Фиксированный макет располагается по центру окна браузера, а его ширина ограничивается заданными размерами в пикселах.

При создании **резинового макета** задается в процентах ширина колонок таким образом, что макет занимает всю свободную ширину окна браузера.

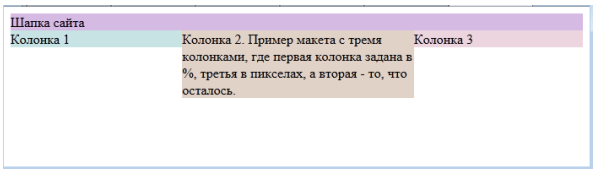
Эластичный макет по своему виду может не отличаться от фиксированного или резинового макета. Размер элементов задаётся в em, привязанных к размеру шрифта. Значение em можно использовать не для всех элементов, оставляя ширину некоторых фиксированной.

Адаптивный макет подстраивается под разрешение монитора и окна браузера, меняя при необходимости ширину макета, число колонок, размеры изображений и текста. Для этого заготавливается несколько стилевых правил или файлов под разный диапазон разрешений, выбор правил происходит через скрипты или CSS3, которые и определяют нужную для этого информацию о пользователе.

Комбинированный макет предполагает использование разной ширины для отдельных частей страницы, например, шапку и подвал делают резиновыми, а контент фиксированным.

Пример макета с тремя колонками, где первая колонка задана в %, третья в пикселах, а вторая – то, что осталось.

```
<html>
<head>
<style>
.header { background: #D5BAE4; }
.layout { position: relative;}
.layout DIV { position: absolute; }
.col1 { background: #C7E3E4; width: 30%; }
.col2 { background: #E0D2C7; left: 30%; right: 200px; }
.col3 { background: #ECD5DE; right: 0; width: 200px; }
</style>
<title>Три колонки</title>
</head>
<body>
<div class="header">Шапка сайта</div>
<div class="layout">
<div class="col1">Колонка 1</div>
<div class="col2">Колонка 2. Пример макета с тремя колонками, где первая колонка задана в %, третья – в пикселах, а вторая – то, что осталось.</div>
<div class="col3">Колонка 3</div>
</div>
</body>
</html>
```



Параметры CSS, управляющие положением на странице:

position: absolute | fixed | relative | static | inherit естанавливает способ позиционирования элемента относительно окна браузера или других объектов на Web-странице.

absolute указывает, что используются абсолютные координаты.

relative указывает, что используются относительные координаты.

inherit наследует значение родителя.

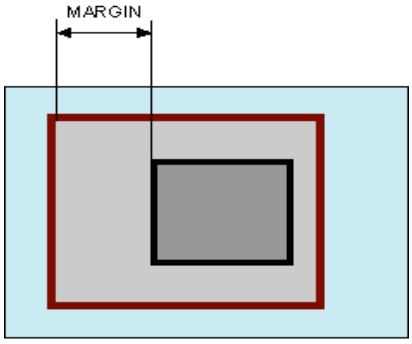
left задает положение относительно левого края контейнера.

top задает положение относительно верхнего края контейнера. Задаются в процентах или пикселях.

z-index указывает на то, какой элемент должен располагаться выше при перекрытии. Измеряется в единицах.

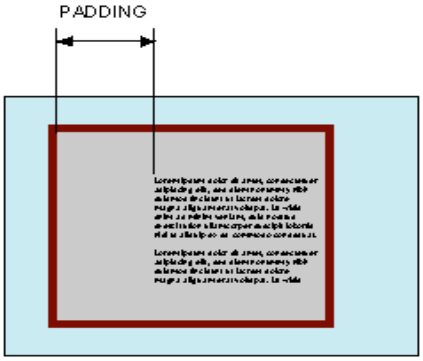
visibility определяет видимость элемента. Принимает значения visible (видимый), hidden (скрытый), inherit (наследуется от родительского элемента).

margin устанавливает величину отступа от каждого края элемента. Отступом является пространство от границы текущего элемента до внутренней границы его родительского элемента.



Разрешается использовать одно, два, три или четыре значения, разделяя их между собой пробелом. Если задано одно значение, то будут установлены четыре одинаковых отступа. Если два, то первое значение - отступ сверху и снизу, второе - слева и справа. Если определены три значения, то первое значение задает отступ от верхнего края, второе - одновременно от левого и правого края, а третье - от нижнего края. Четыре параметра определяют отступ от верхнего, правого, нижнего и левого края.

padding устанавливает значение полей вокруг содержимого элемента. Полем называется расстояние от внутреннего края рамки элемента до воображаемого прямоугольника, ограничивающего его содержимое. Так же как и у margin возможны несколько вариантов значений.



2.5. Основы языка PHP

Особенности серверного программирования

Серверные программы позволяют решать ряд важных задач, которые невозможно решить с помощью базовых технологий разработки Web-страниц и клиентского программирования. К таким задачам относится, например, сохранение полученной от пользователей информации, взаимодействие с внешней базой данных и т. д. Поэтому ни одно сколько-нибудь серьезное Web-приложение не обходится без серверной части.

Главной особенностью серверных программ является то, что они могут быть выполнены только на сервере, в отличие от HTML, CSS и сценариев JavaScript, которые интерпретируются пользовательским клиентом - браузером. Обращение браузера к Web-серверу может осуществляться и в рамках одного компьютера, в этом случае установка и функционирование Web-сервера может использоваться для отладки серверных программ. Для

того чтобы гарантировать доступ к данным на компьютере для других пользователей, одного Web-сервера недостаточно, потребуются также выделенный IP-адрес и запись в сервере DNS (чтобы пользователи имели возможность обращаться к серверу по символическому имени, а не только по IP-адресу).

Поэтому для разработки и отладки серверной части Web-приложения необходимо развернуть и настроить на своем компьютере программное обеспечение Web-сервера. Кроме того, необходимо установить транслятор языка серверного программирования, систему управления базой данных (СУБД), с которой будет взаимодействовать web-приложение, и обеспечить взаимодействие всех этих компонентов. Для облегчения этого процесса часто используют устоявшиеся связки – комплексы серверного программного обеспечения:

– Web-сервер Apache, язык PHP, СУБД MySQL;

– Web-сервер MS IIS, ASP.NET, СУБД MS SQL Server.

Это не значит, например, что Web-сервер MS IIS не может работать с PHP, или Web-сервер Apache работает только с PHP, или что невозможно организовать взаимодействие скриптов PHP с SQL-сервером от Microsoft, однако внутри связки такое взаимодействие отработано и, как правило, не требует дополнительной настройки. Кроме того, предлагаются готовые пакеты, включающие все основные компоненты связки (и, возможно, некоторые дополнительные), обеспечивающие их интеграцию, а также предоставляющие интерфейс (оболочку) для управления. Наиболее известные пакеты для связки Apache-PHP-MySQL – XAMPP, Denwer, EasyPHP.

Альтернатива – разработка с использованием готовых систем управления контентом (CMS), таких как Drupal, Joomla!, WordPressMS, MS WebMatrix и др., не требующих знания языков программирования, что является их основным преимуществом. В то же время следует помнить, что каждая CMS-система имеет свои ограничения, что может затруднить или сделать невозможным ее настройку под решение конкретной (в особенности не типовой) задачи.

При выборе инструментов разработки Web-приложения приходится сравнивать не просто отдельные языки программирования, но и технологии, обеспечивающие взаимодействие с базами данных, работу на стороне клиента и сервера, то есть всю связку, а также возможности технической поддержки, стоимость программного обеспечения и сопровождения, масштабы Web-приложения, вопросы безопасности и т.д. Иногда требуется разработать приложение для конкретной платформы, в этом случае выбор технологии серверного программирования ограничен тем набором языков, которые поддерживаются Web-сервером. Если стоит задача обеспечения переносимости между платформами, лучше остановить выбор на наиболее популярных серверных технологиях, поддерживаемых большинством платформ. К таким технологиям относится язык PHP, являющийся одним из лидеров среди языков разработки динамических Web-сайтов благодаря своей простоте, скорости выполнения, богатой функциональности, кроссплатформенности и открытому исходному коду (Open Source),

распространяющемуся под собственной лицензией.

Остановимся на связке Apache-PHP-MySQL и оболочке XAMPP, предоставляющей визуальный интерфейс для управления компонентами связки (рис. 10), в том числе и в среде ОС Windows.

Полный пакет XAMPP содержит:

– web-сервер Apache с поддержкой SSL;

– СУБД MySQL;

– язык PHP;

– язык;

– FTP-сервер FileZilla;

– POP3/SMTP сервер;

– утилиту phpMyAdmin для визуального управления базами данных MySQL.

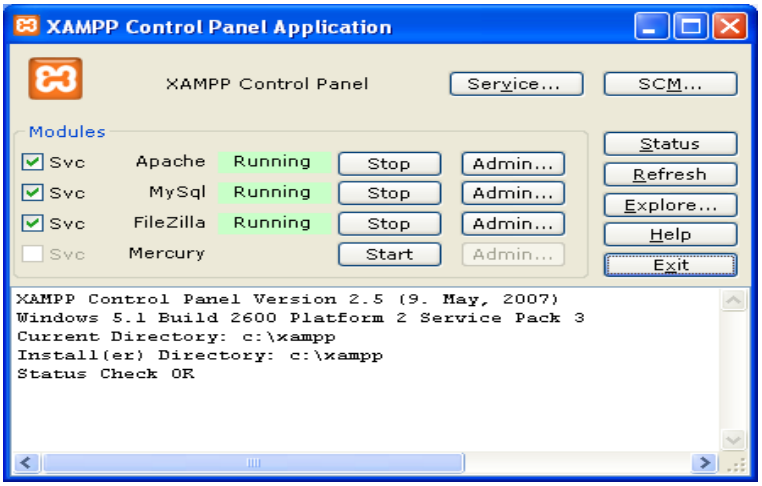


Рис. 10. Панель управления XAMPP

Для простоты работы некоторые возможности и настройки безопасности в XAMPP по умолчанию отключены, и в целом его рекомендуется использовать только в достаточно дружественном окружении, например, для отладки серверных программ в процессе разработки на локальном компьютере.

После запуска XAMPP, служб web-сервера (и MySQL) можно перейти в Web-интерфейс XAMPP, набрав в адресной строке браузера: http://localhost или http://127.0.0.1. Web-интерфейс XAMPP содержит ссылки для отображения текущего статуса и настроек компонент XAMPP (*Status* - рис. 10, *phpinfo()*, *perlinfo()*), документацию и примеры, обеспечивает переход в оболочку phpMyAdmin, к почтовому и FTP-серверу, а также позволяет задавать настройки безопасности. Меню *Security* служит для просмотра статуса безопасности компонент XAMPP. Для задания пароля администратора СУБД MySQL и пароля папки XAMPP для ограничения доступа к ней из локальной сети следует перейти по ссылке http://localhost/security/xamppsecurity.php (рис. 11).

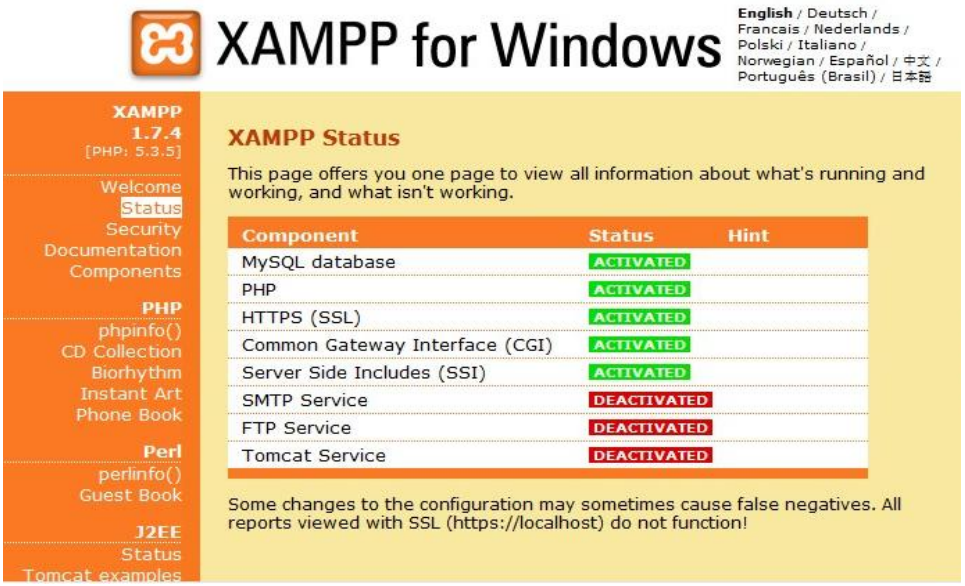


Рис.11. Просмотр статуса компонент XAMPP

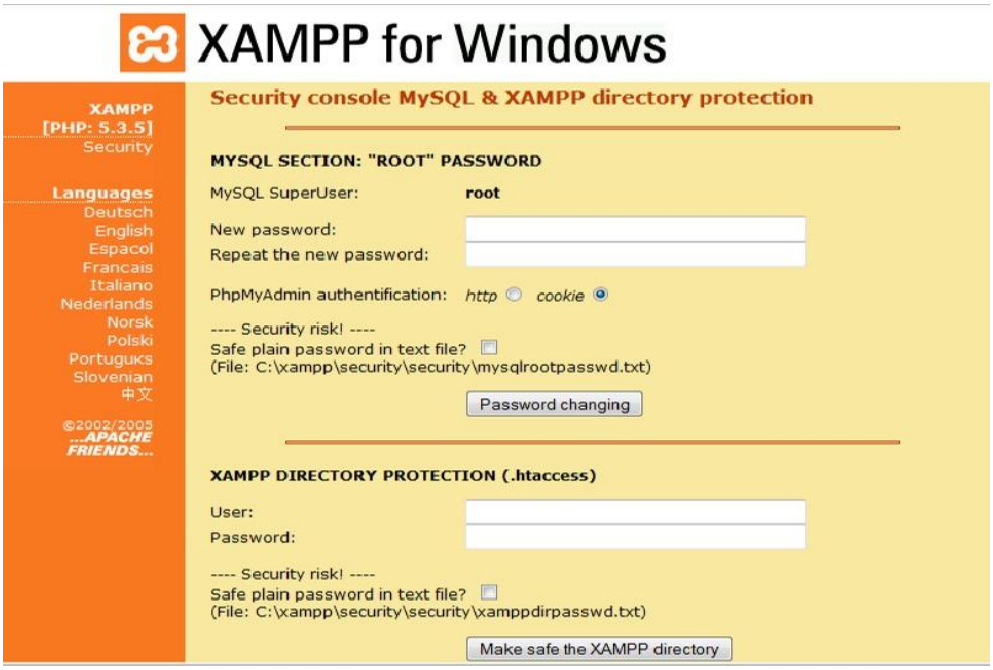


Рис.12. Настройка параметров безопасности XAMPP

Для того чтобы создать новый сайт на локальном Web-сервере, необходимо создать папку с названием сайта (например, site) в папке \xampp\htdocs\ и разместить в ней страницы сайта. Файл домашней страницы должен иметь имя index с расширением htm, html или php. Обращение к локальному Web-сайту в этом случае будет иметь вид http://localhost/site.

Если необходимо хранить файлы Web-сайта в другой папке, может быть создан виртуальный хост. По умолчанию папка XAMPP создается в корне системного диска C:, назначения подпапок XAMPP и местонахождение некоторых важных файлов его компонент приведено в табл. 28.

Таблица 28. Назначения некоторых подпапок и файлов XAMPP

\xampp\htdocs	Папка для размещения локальных сайтов
\xampp\php	Папка транслятора PHP
\xampp\mysql\data	Папка баз данных MySQL
\xampp\cgi-bin	Папка для CGI-программ
\xampp\anonymous	Папка для анонимного доступа через FTP (пользователь anonymous)
\xampp\apache\logs\error.log	Лог-файл с описанием ошибок сервера Apache
\xampp\mysql\data\mysql.err	Лог-файл с описанием ошибок СУБД MySQL
\xampp\php\php.ini	Файл настроек языка PHP

Основы синтаксиса языка PHP

PHP - скриптовый язык программирования, созданный для генерации динамических страниц на web-сервере и работы с базами данных. PHP поддерживается подавляющим большинством хост-провайдеров. PHP является интерпретируемым языком и устанавливается с Web-сервером Apache как расширение сервера. Совместная работа PHP-интерпретатора с Web-сервером MS IIS осуществляется через CGI-интерфейс.

Сценарии PHP встраиваются непосредственно в Web-страницу и интерпретируются на сервере перед отправкой страницы пользователю. Пользователь получит страницу с результатом выполнения сценария, но не увидит сам PHP-код.

Встраивание сценария в HTML/XHTML-код Web-страницы производится с помощью специальных тегов:

```
<?php
...
?>
```

Существует сокращенная форма этого тега, которая доступна, если в файле настроек языка PHP (php.ini) включен параметр short_open_tag (short_open_tag = On):

```
<?
...
?>
```

Некоторые Web-сервера пропускают через PHP-интерпретатор не только страницы с расширением php, а все страницы сайта (с расширениями htm и html). Интерпретация обычной Web-страницы, не содержащей PHP-инструкций, никак не изменит ее внешний вид. Другие сервера интерпретируют только те страницы, которые имеют расширение php. Поэтому рекомендуется в обязательном порядке менять расширение web-страниц на php при добавлении в них PHP-кода. Следует отметить, что файл с расширением php не обязательно содержит обычный HTML/XHTML-код.

Например, если разместить в папке сайта (site) на сервере файл index.php с кодом:

```
<?php
echo ("Hello, World!");
```

?>

Тогда при обращении к сайту через сервер (например, по адресу

http://localhost/site), браузер выведет страницу с приветствием (рис. 13).

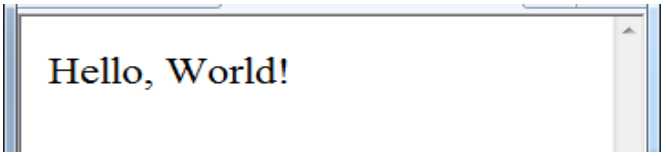


Рис.13. Результат выполнения простого PHP-сценария

В то же время, если открыть страницу index.php в браузере без посредничества сервера, напрямую с диска, будет выдан текст PHP-сценария (рис. 14).

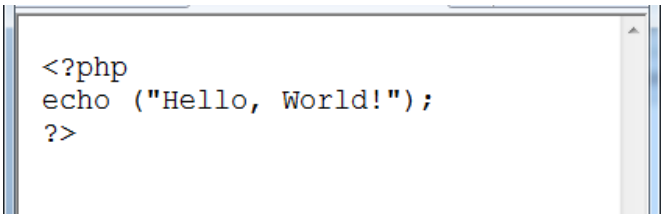


Рис.14. Отображение PHP-сценария без посредничества сервера

В PHP-выдачу могут быть добавлены теги HTML для форматирования текста, например:

<?php

echo ("Hello, World!");

?>

После интерпретации сервером текст приветствия будет отображен курсивом.

Сценарий PHP может быть сохранен в отдельном файле с расширением php, а затем вызван из другого сценария функцией include(), например: include("db_info.php");.

Возможность подключения к сценарию внешних файлов обеспечивают также функции require(), include_once() и require_once(). Функции require() и include() отличаются лишь реакцией на невозможность получения запрошенного ресурса. В случае недоступности запрошенного ресурса функция include() выводит предупреждение и пытается продолжить исполнение сценария, а функция require() при недоступности ресурса останавливает обработку. Функцию require() или require_once() рекомендуется использовать, если подключается файл с определениями критически важных функций или переменных (например, описывающих подключение к базе данных), без которых сценарий не сможет функционировать.

При подключении сценариев с многоступенчатой вложенностью сценарий может подключиться повторно, что может привести к возникновению ошибок (например, повторное определение пользовательской функции). Исключить подобные ошибки позволяет использование include_once() или require_once(), которые проверяют, был ли сценарий уже подключен ранее.

Синтаксис языка PHP во многом схож с синтаксисом JavaScript.

Каждый оператор заканчивается знаком точки с запятой (;). Блоки операторов заключаются в фигурные скобки { }.

Функции определяются так же, как и в JavaScript, с помощью ключевого слова function; для возврата значения из функции используется ключевое слово return. Имена функций в PHP не чувствительны к регистру символов.

Комментарий предваряется знаками // или #. Можно также использовать многострочные комментарии в стиле языка Си, заключенные в символы /* ... */.

Константы объявляются с помощью функции define(), например: define("const_name","Зима"); или define("value1",751);. Проверка того, определена ли константа, производится с помощью функции defined(), принимающей логические значения (true, false). В выражениях константы указываются по имени без двойных кавычек.

Переменные предваряются знаком доллара ($), за которым следует идентификатор – последовательность букв, цифр и символов подчеркивания, начинающаяся с буквы. Так же как и в JavaScript, имена переменных в PHP чувствительны к регистру символов.

PHP не требует явного объявления переменных и указания их типа перед использованием, тип переменной определяется ее значением, при этом в процессе выполнения сценария PHP переменная может менять свой тип. Существует возможность явного определения/переопределения типа данных (integer, float, double, real, srting, boolean, array, object) с помощью функции settype; просмотр типа переменной осуществляется функцией gettype.

Пример:

```
<?php
$my_var="3";
echo("Исходный тип переменной ".gettype($my_var));
settype($my_var,"integer");
echo("<br>Новый тип переменной ".gettype($my_var));
?>
```

Результат выполнения данного сценария приведен на рис. 15. В примере = означает оператор присваивания, а знак точки (.) - операцию конкатенации строк.

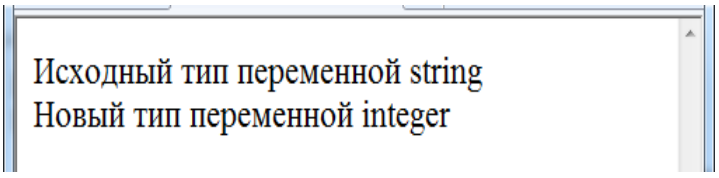


Рис.15. Изменение типа переменной в PHP-сценарии

Переопределение типа переменной может производиться указанием нового типа в круглых скобках перед именем переменной, например:

```
$my_var=(int)$my_var;
```

Константы, как только они определены, всегда видимы глобально, то есть могут использоваться как внутри, так и вне функций.

Областью видимости *локальных* переменных является функция, внутри которой они определены. Время жизни локальной переменной определяется временем выполнения соответствующей функции или сценария. Локальная

переменная при каждом вызове функции инициализируется заново. Объявление статической переменной позволяет сохранить ее значение между вызовами функции.

Глобальные переменные определены в сценарии, видны из любого места сценария, но не внутри функций. Доступ из функции к переменным сценария можно получить через массив $GLOBALS.

В следующем примере в теле функции change_my_var() через массив $GLOBALS идет обращение к переменной, определенной вне функции, то есть внешней по отношению к change_my_var().

```
<?php
function change_my_var()
{
$GLOBALS["my_var"]=5;
echo("Значение из функции: ".$GLOBALS["my_var"]);
}
$my_var=3;
echo("Исходное значение переменной: $my_var<br />");
change_my_var();
?>
```

Вывод этого сценария приведен на рис. 16.

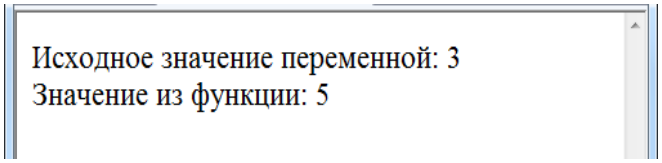


Рис.16. Изменение значения внешней переменной из функции

Переменную внутри функции можно сделать глобальной, указав перед ней ключевое слово global, например: global $my_var;. Переменные, использованные внутри функции, которые объявлены как глобальные, ссылаются на глобальные переменные с теми же именами.

Когда функции передается аргумент, создается его локальная копия для хранения значения. Изменение этого значения касается только локальной копии переменной в функции, никак не отражаясь на источнике параметра. Однако можно определять *передачу параметров по ссылке*, изменение которых будет отражаться на исходной переменной. Для этого в описании функции перед именем параметра помещают символ амперсанда (&).

Внешние переменные, то есть полученные из окружения PHP-интерпретатора (от браузера или от сервера), являются *суперглобальными*, то есть доступны как любому сценарию, так и внутри любой функции.

В PHP при помощи ссылки можно создать две переменные, которые будут ссылаться на одно и то же содержимое. Для создания ссылки-переменной в правой части оператора присваивания перед именем переменной следует указать символ амперсанда (&), например:

```
$var1=&$var2;
```

После этого изменение значения одной из переменных автоматически приведет к изменению значения другой. Следует отметить, что использование ссылок-переменных иногда существенно осложняет поиск

ошибок в коде.

Оператор присваивания обозначается знаком =; как и в JavaScript, может быть использована сокращенная запись оператора присваивания: +=, -=, *= и т.д.

Арифметические операции: +, -, *, /, % (остаток целочисленного деления), ++ (инкремент), -- (декремент). Существует префиксная и постфиксная формы записи инкремента и декремента. В префиксной форме сначала производится инкремент/декремент, затем вычисляется выражение, в постфиксной – сначала вычисляется выражение, затем инкремент/декремент.

Конкатенация строк обозначается знаком точки (.).

Операторы сравнения: <, >, <=, >=, == (равенство), != (неравенство), === (идентичность), !== (неидентичность).

Логические операторы: and, && (логическое И), or, || (логическое ИЛИ), ! (отрицание НЕ), xor (исключающее ИЛИ). Для логического И, ИЛИ имеется два варианта обозначения, которые различаются приоритетом выполнения.

Побитовые операции: & (побитовое И), | (побитовое ИЛИ), ^ (побитовый XOR), ~ (побитовое НЕ), << (сдвиг влево на указанное число бит), >> (сдвиг вправо).

Приоритет выполнения операторов (в порядке убывания): ++, --, !, ~, (унарный минус), преобразования типа, *, /, %, +, -, . (конкатенация), <<, >>, <, <=, >, >=, ==, !=, ===, !==, &, ^, |, &&, ||, ? : (условный оператор), = (присваивание), and, or, xor.

В PHP следующие управляющие конструкции: условный оператор, условная операция, оператор выбора, операторы цикла, операторы выхода из цикла.

Условная конструкция имеет такой вид:

if (логическое выражение)

{ операторы1}

else

{ операторы2}

Допустима сокращенная форма записи условного оператора без части else.

Конструкция if … else … может быть заменена условной операцией, имеющей следующий синтаксис:

выражение1 ? выражение2 : выражение3

Выражение1 – логическое выражение, истинность которого проверяется; результатом будет выражение2, если выражение1 истинно, и выражение3 - в противном случае.

Пример - вычисление абсолютного значения переменной $x:

$x<0 ?-$x:$x;

Другой пример – задание текстового значения переменной в зависимости от выполнения условия:

$ban=($login==true)?"Добро пожаловать!":"Зарегистрируйтесь!"

Условная конструкция с elseif позволяет проверять дополнительные условия, пока не будет найдено истинное или достигнут блок else. У каждой инструкции elseif есть собственный блок кода, размещаемый непосредственно после условного выражения инструкции elseif. Инструкция

elseif идет после инструкции if и перед инструкцией else, если таковая имеется.

В общем случае синтаксис конструкции elseif имеет такой вид:

```
if (логическое выражение1) {
операторы1;
}
elseif (логическое выражение2) {
операторы2;
}
elseif (логическое выражение3) {
операторы3;
}
… else {
операторыN;
}
```

Конструкция выбора switch полезна, если требуется выполнять различные действия в зависимости от различных значений переменной. Синтаксис конструкции switch:

```
switch(выражение) { case значение1: операторы1;
break;
case значение2:

операторы2; break;
…
case значениеN: операторыN; break;
default: операторы по умолчанию; break;
}
```

Конструкция switch имеет свои особенности. Если найдено соответствие текущего значения выражения (переменной) значению, определенному в каком-либо из блоков case, будет исполнен соответствующий блок операторов, а также все расположенные ниже блоки кода до конца инструкции switch или до ключевого слова break.

Если ни одного соответствия не найдено, исполняется блок default. Блок default не является обязательным, он может отсутствовать. Также не обязательно указание операторов break, которые осуществляют выход из конструкции выбора после нахождения совпадения.

Указание операторов break позволяет выполнять действия, предусмотренные только одной альтернативой. Если операторы break в конструкции выбора отсутствуют, будут выполнены действия, предусмотренные найденной и всеми последующими альтернативами.

Пример конструкции выбора без операторов break: switch ($action) {

```
case "assemble":
echo ("Укомплектовать заказ<br />"); case "pack":
echo ("Упаковать заказ<br />"); case "send":
echo ("Отправить заказ<br />");
}
```

В данном примере, если переменная $action примет значение «assemble», сценарий выдаст три строки (рис. 17).

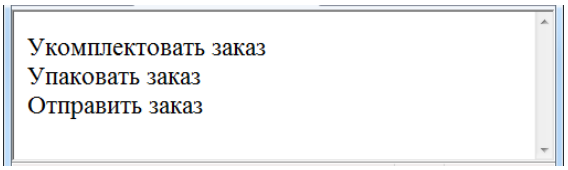


Рис.16. Вывод конструкции выбора без операторов switch

Если же значением переменной $action оказалось "pack", будут выданы две строки:

Упаковать заказ Отправить заказ

В то же время конструкция выбора с операторами break всегда выдавала бы одну строку, точно соответствующую значению "assemble" или значению "pack".

Для организации циклов в PHP используются конструкции while (цикл с предусловием), do … while (цикл с постусловием), for и foreach.

Синтаксис цикла while:

while (условие выполнения цикла)

{

операторы

}

Синтаксис цикла do…while: do

{

операторы

} while (условие выполнения цикла)

Оператор for определяет цикл с заданным числом повторений и имеет синтаксис:

for (выражение1;выражение2;выражение3)

{

операторы

}

Выражение1 – выражение инициализации счетчика цикла (например, $i=0); выражение2 – условие выполнения цикла (например, $i<5); выражение3 – выражение, задающее изменения счетчика (например, $i++).

Цикл foreach предназначен для работы с массивами.

Для выхода из цикла используется оператор break, для завершения текущей итерации и перехода к следующей – оператор continue.

Оператор exit прекращает выполнение оставшейся части сценария.

Управляющие конструкции в PHP поддерживают альтернативный синтаксис, в котором открывающая фигурная скобка ({) заменяется знаком двоеточия (:), а закрывающая фигурная скобка (}) – закрывающим ключевым словом (endif, endswitch, endwhile, endfor или endforeach в зависимости от управляющей конструкции).

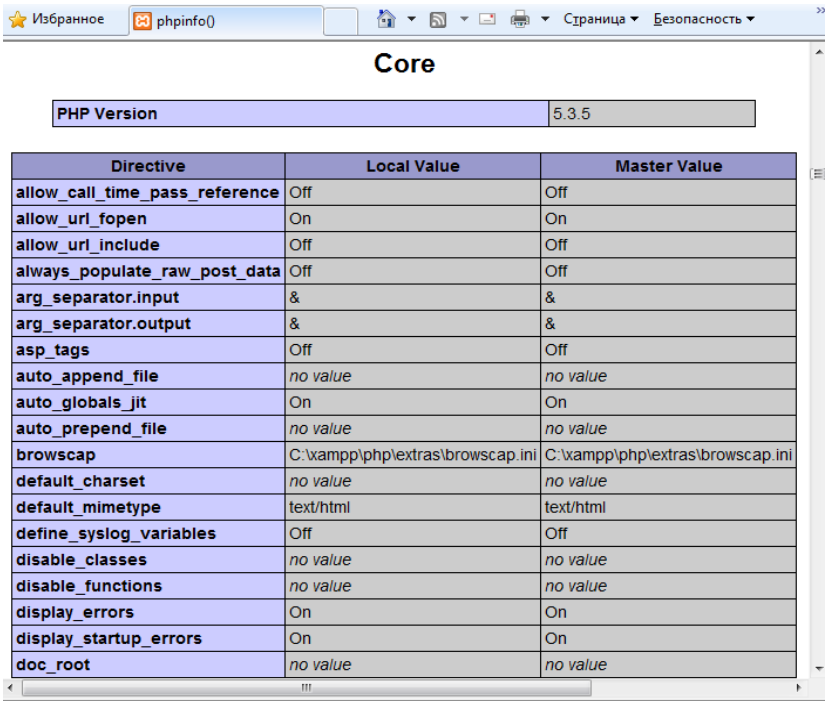
Обработка данных Web-форм

Для того чтобы серверный сценарий на языке PHP мог обрабатывать данные, введенные пользователем в поля Web-формы, URL-адрес этого сценария должен быть указан в качестве значения параметра action тега формы form. Способ обработки данных формы зависит от метода отправки

данных (get или post) и настроек PHP.

Просмотр текущих настроек PHP можно осуществить из PHP–сценария с помощью функции phpinfo() или с помощью одноименной команды оболочки XAMPP (рис. 18).

Важно, что значения параметров PHP чувствительны к регистру символов.



Core

PHP Version	5.3.5

Directive	Local Value	Master Value
allow_call_time_pass_reference	Off	Off
allow_url_fopen	On	On
allow_url_include	Off	Off
always_populate_raw_post_data	Off	Off
arg_separator.input	&	&
arg_separator.output	&	&
asp_tags	Off	Off
auto_append_file	*no value*	*no value*
auto_globals_jit	On	On
auto_prepend_file	*no value*	*no value*
browscap	C:\xampp\php\extras\browscap.ini	C:\xampp\php\extras\browscap.ini
default_charset	*no value*	*no value*
default_mimetype	text/html	text/html
define_syslog_variables	Off	Off
disable_classes	*no value*	*no value*
disable_functions	*no value*	*no value*
display_errors	On	On
display_startup_errors	On	On
doc_root	*no value*	*no value*

Рис.18. Просмотр настроек PHP командой phpinfo()

Если в файле настроек php.ini включен параметр register_globals (register_globals = On), то при отправке данных с помощью методов get и post автоматически создаются переменные с глобальной областью видимости. При этом имена переменных совпадают с именами передаваемых полей формы.

Пусть в корневом каталоге сайта размещены два файла: файл index.html с простой web-формой и файл сценария 1.php.

В теге формы обязательно должны быть определены метод отправки данных и сценарий, которому будут переданы данные формы:

```
<form name="form1" method="get" action="1.php">
Здравствуйте, как Вас зовут?<br />
<input type="text" name="text1" id="fio" /><br />
<br />
<input type="submit" value="Отправить данные" />
</form>
```

Тогда при включенном параметре register_globals после отправки формы сценарий 1.php имеет доступ к переменной $text1, содержащей введенный пользователем текст.

Пример текста сценария для обработки данных простой формы (файл 1.php):

```
<?php
echo ("<p><b>Добро пожаловать, <br /> $text1 </b></p>"); ?>
```

В результате отправки формы пользователю будет выдана страница приветствия. На рис. 19 показан результат выдачи серверного сценария для случая, когда пользователь ввел в поле формы текст «гость».

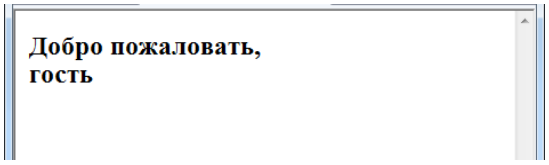


Рис. 19. Пример обработки данных формы серверным сценарием

Рассмотренный прямой доступ к автоматически создаваемым глобальным переменным очень удобен, однако реальны проблемы с безопасностью в связи с возможностью проверки источника значения переменной перед его использованием. Если параметр register_globals включен, перед выполнением кода инициализируются различные внешние переменные, при этом переменные, определяемые разработчиком внутри скрипта, и передаваемые пользователем внешние данные могут перемешиваться. Опасность заключается в том, что PHP не требует предварительного объявления переменной, и это позволяет злоумышленнику вызвать PHP-сценарий с произвольными get- или post-параметрами, которые не предусмотрены формой.

Например, в описанном выше примере можно напрямую, минуя страницу с формой, вызвать результат выполнения серверного сценария (в предположении, что он расположен в папке site на локальном сервере), набрав в адресной строке браузера:

http://localhost/site/1.php?text1=test

В данном примере никаких проблем с безопасностью не возникнет, однако если имя переданного таким образом параметра совпадет с именем какой-то важной переменной (например, содержащей результат процедуры прохождения аутентификации), то злоумышленник сможет повлиять на функциональность Web-приложения, добавив ложный параметр.

Решение данной проблемы заключается в инициализации важных переменных до начала их использования, а также извлечение значений с помощью методов, которые однозначно указывают на их происхождение, чтобы избежать непроверенных значений, которые поступают непосредственно от пользователя. Поэтому был предложен другой способ обработки внешних переменных.

В поздних версиях языка PHP (начиная с версии 4.1.0) параметр register_globals по умолчанию выключен (register_globals = Off), а доступ к значениям внешних переменных осуществляется через суперглобальные массивы:

– $_GET - массив переменных, переданных в сценарий из web-формы методом get;

– $_POST - массив переменных, переданных в сценарий из web-формы методом post;

– $_COOKIES - массив cookie-переменных;

– $_REQUEST - массив пользовательского ввода, включая содержимое массивов $_GET, $_POST и $_COOKIES;

– суперглобальный массив $_REQUEST позволяет получать предоставляемые пользователем значения, не заботясь об их происхождении;

– $_SERVER - массив переменных среды сервера;

– $_FILES - массив переменных, связанных с загрузкой файлов;

– $_ENV - массив переменных окружения;

– $_SESSION - массив переменных сеанса;

– $_GLOBALS - массив всех глобальных переменных.

В предположении, что параметр register_globals в настройках PHP выключен, текст сценария 1.php из примера выше примет вид:

```
<?php
echo ("<p><b>Добро пожаловать, <br />". $_GET["text1"]. "</b></p>");
?>
```

Старый же сценарий перестанет работать (рис. 20), поскольку переменной $text1 теперь не существует.

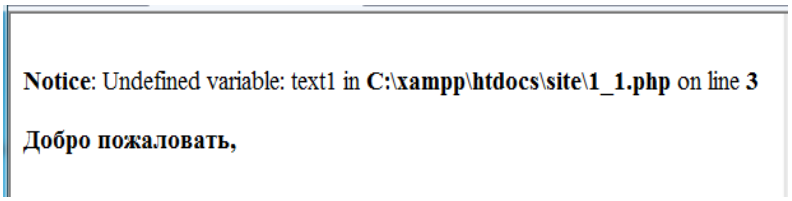


Рис. 20. Сообщение об ошибке после отключения параметра register_globals

Иногда возникает необходимость обеспечить работу старых сценариев, написанных в предположении включенного параметра register_globals, не жертвуя при этом уровнем безопасности. Такая задача может возникнуть, например, при использовании готовых PHP-сценариев, находящихся в открытом доступе в Интернете. Данную проблему можно решить, подключив сценарий с переопределением переменных через суперглобальные массивы. Например, к старому сценарию 1.php можно подключить сценарий old.php следующего содержимого:

```
<?php
$text1 = $_GET["text1"];
?>
```

Текст старого сценария 1.php, написанного в предположении включенного параметра register_globals, тогда может быть модифицирован следующим образом:

```
<?php include("old.php");
echo ("<p><b>Добро пожаловать, <br /> $text1 </b></p>");
?>
```

Возможна также предварительная проверка существования переменной перед использованием логической функцией isset().

В дальнейшем в примерах серверных сценариев обработка данных форм производится через суперглобальные массивы в предположении выключенного параметра register_globals.

Даже при выключенном в целях повышения безопасности параметре register_globals сохраняется необходимость проверки корректности введенных пользователем данных.

Заполнение поля формы можно проверить с помощью логической функции empty(), которая возвращает значение 1 (true) в том случае, если переменной не существует или она имеет пустое значение.

Пример проверки заполнения поля text1 перед выдачей приветствия сценарием 1.php:

```
<?php
if (empty($_GET["text1"]))
echo ("<p><b>Добро пожаловать, <br /> неизвестный друг</b></p>");
else
echo ("<p><b>Добро пожаловать, <br />". $_GET["text1"]. "</b></p>");
?>
```

PHP поддерживает работу с регулярными выражениями, что позволяет проверять пользовательские данные на соответствие шаблону. Построение регулярных выражений подробно описано в предыдущем параграфе, посвященном работе с JavaScript.

В PHP есть набор функций с именами, начинающимися на preg_, которые осуществляют операции с регулярными выражениями. Эти функции принимают в качестве аргумента регулярное выражение в виде строки и выполняют операции над строками. Чаще всего используется функция preg_match(), которая выполняет поиск в строке всех совпадений по заданному регулярному выражению (шаблону) и возвращает значение true, если совпадение было найдено. Функция preg_match() может также выдавать массив найденных совпадений. Если совпадения не найдено, массив останется пустым.

В общем случае функция preg_match() имеет следующий синтаксис:

preg_match (шаблон, строка, массив совпадений)

Последний параметр (массив совпадений) является необязательным.

Рассмотрим пример использования шаблонов для проверки данных формы на стороне сервера. В примере используется страница с Web-Формой для ввода имени, пола и телефона пользователя. Данные формы передаются серверному сценарию с помощью метода post. Процедура проверки данных формы должна проверять заполнение текстовых полей для ввода имени (fio) и телефона (tel), а также соответствие введенного номера телефона заданному шаблону: (XXX)XXX-XX-XX, где X – цифры. Шаблон для проверки номера телефона с помощью регулярных выражений формируется так же, как и для сценария JavaScript, и имеет вид:

/^[(][0-9]{3}[)][0-9]{3}(-)[0-9]{2}(-)[0-9]{2}$/.

Тогда серверная процедура проверки может быть следующей:

```
<?php
$tel_num="/^[(][0-9]{3}[)][0-9]{3}(-)[0-9]{2}(-)[0-9]{2}$/";
if (empty($_POST["fio"]) || empty($_POST["tel"])) echo ("<p>Заполнены не все поля</p>");
elseif (preg_match($tel_num, $_POST["tel"]))
echo ("<p>Спасибо, ваши данные успешно прошли проверку</p>");
//Здесь должна располагаться процедура дальнейшей
//обработки данных
else
echo("<p>Неправильный номер телефона!<br /> Введите номер телефона
```

в формате
 (XXX)XXX-XX-XX");
?>

Рассмотренный сценарий фактически не производит с пользовательскими данными никаких действий, кроме проверки. В зависимости от поставленной задачи процедура дальнейшей обработки данных может сохранять их во внешнем файле, либо заносить в базу данных, либо формировать новую Web-страницу на основе пользовательских данных, например, с извлеченной из базы данных информацией.

Прежде чем рассмотреть реализацию возможных процедур обработки, следует еще поговорить о проверке полученных от пользователя данных в контексте обеспечения безопасности Web-приложения. Речь идет об атаках межсайтового скриптинга (cross-site scripting, XSS) и SQL- инъекции.

Межсайтовый скриптинг предполагает использование злоумышленником Web-приложения, для того чтобы передать исполняемый исходный код другому пользователю, применяемому данное приложение. Эта атака становится возможной, если сценарий отображает пользовательские данные без их предварительной проверки. Злоумышленник может ввести в поле формы теги, код на языке JavaScript, другой исполняемый код (Ac- tiveX (OLE), VBscript, Flash и т.п.).

Для предотвращения подобного типа атак старые версии интерпретатора PHP имели встроенный механизм «Волшебные кавычки» (Magic Quotes), обеспечивающий автоматическое экранирование входящих данных PHP-скрипта. Начиная с версии PHP 5.4.0 данный механизм не поддерживается. Проверить поддержку «Волшебных кавычек» можно с помощью логической функции get_magic_quotes_gpc(): true - механизм включен, false - механизм отключен или не поддерживается.

Экранирование специальных символов, таких как угловые скобки (<, >), двойные кавычки (") и др., путем их конвертирования в соответствующие мнемоники HTML (< , > , " и др.) осуществляется функцией htmlentities().

Существуют сотни различных вариантов атак, использующих XSS, включая атаки, которые не применяют символы угловых скобок (<, >). Поэтому фильтрация не всегда эффективна, для определения легитимности введенных данных рекомендуется обязательно сравнивать их с набором разрешенных значений (например, с помощью шаблона на основе регулярных выражений).

SQL-инъекция – это прямое внедрение вредоносных инструкций в SQL-запросы с целью отображения скрытых данных, их изменения или выполнения вредоносного кода на сервере базы данных. Атака может быть реализована, если приложение строит SQL-запросы из пользовательского ввода и статических параметров.

Поэтому перед отправкой весь пользовательский ввод в базу данных следует пропускать его через функцию mysql_real_escape_string().

Другая возможная атака – подключение файла, определенного пользовательским значением без предварительной проверки (*php-include*). Речь идет о функциях include() или require(). Функция include() будет исполнять только ту часть файла, которая заключена между специальными

тегами, обозначающими начало (<?php) и конец (?>) PHP-сценария. Если подключаемый файл имеет содержимое, не заключенное в эти теги, оно будет выдано в текстовом виде.

Когда подключаемый к сценарию файл определяется переменной, значение которой получено от пользователя, злоумышленник может ввести произвольное имя файла, например, файла, предназначенного для хранения паролей, и просмотреть его содержимое.

Помимо доступа к файлам, находящимся на диске сервера, функция include() также предоставляет возможность запускать скрипты, находящиеся на удаленных рабочих станциях.

Для предотвращения подобных атак рекомендуется жестко определять все допустимые альтернативы подключаемых файлов, например, с помощью структуры выбора switch, а также ограничивать доступ к папкам с важными файлами с помощью .htaccess.

Организация взаимодействия с базой данных

Несмотря на то, что язык PHP имеет ряд функций для манипулирования внешними файлами, непосредственная работа с файлами из PHP-сценария всегда несет в себе угрозы безопасности, поэтому лучшим решением является сохранение информации в базе данных, а не в файлах.

База данных представляет собой структурированную совокупность данных. Система управления базами данных (СУБД) автоматизирует большую часть задач, связанных с хранением и извлечением пользовательской информации на основе заданных критериев. Для записи, выборки и обработки данных, хранящихся в компьютерной базе данных, используется язык структурированных запросов SQL (Structured Query Language).

Хотя PHP поддерживает работу с различными СУБД, обычно разработчики используют СУБД MySQL, имеющую взаимоувязанный с PHP программный интерфейс.

MySQL – реляционная СУБД, являющаяся, как и PHP, бесплатно распространяемой кроссплатформенной системой с открытым исходным кодом. MySQL отличается высокой скоростью работы, масштабируемостью, обеспечивает многопользовательский доступ и поддерживает механизм разграничения доступа.

Сервер MySQL входит в комплект установки XAMPP. Каждый сервер MySQL может содержать несколько баз данных, где группируются таблицы.

Создание пользователей базы данных и назначение им полномочий позволяет ограничить круг пользователей, обладающих правом доступа к таблицам на сервере. В MySQL существует три типа полномочий: полномочия обычных пользователей, полномочия администраторов и специальные полномочия. Полномочия обычных пользователей связаны с определенными командами SQL и правами на их выполнение. Административные полномочия включают доступ к базе данных MySQL, поскольку именно в ней хранятся учетные записи пользователей, пароли.

Если MySQL установлена на том же компьютере, что и Web-сервер с PHP (например, в составе XAMPP), работа с СУБД по умолчанию осуществляется от имени пользователя root. Пользователю root можно задать

пароль. Как правило, в целях безопасности доступ под именем root разрешается только с того компьютера, на котором стоит сервер MySQL. Однако посетителю сайта совсем не обязательно иметь доступ к MySQL для получения информации – за него это будет делать PHP-сценарий, который выполняется на компьютере с Web-сервером. Серверный сценарий предоставляет пользователю не доступ к базе данных, а результат своей работы.

У MySQL есть собственный интерфейс для организации взаимодействия с клиентами. Один из способов взаимодействия основан на использовании командной строки MySQL (MySQL command line client). Для этих целей можно также использовать оболочку phpMyAdmin - инструмент работы с MySQL через web-интерфейс (рис. 21).

Запуск phpMyAdmin производится:

– через адресную строку браузера с помощью следующего URL: http://localhost/phpmyadmin;

– с помощью кнопки Admin в строке MySQL панели управления XAMPP;

– с помощью команды phpMyAdmin в группе Tools Web-оболочки XAMPP.

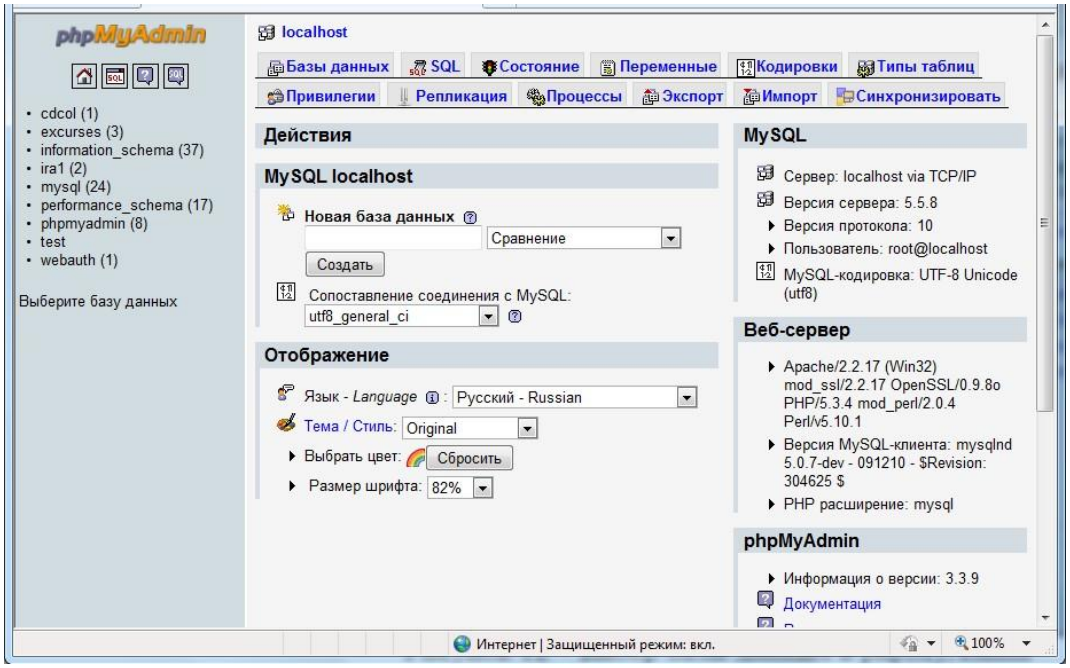


Рис. 21. Стартовое окно оболочки phpMyAdmin

Оболочка phpMyAdmin используется для редактирования и просмотра информации, хранимой в базах данных сервера, она позволяет осуществлять:

– создание новой или выбор существующей базы данных;

– действия с таблицами базы данных (создание, просмотр хранимых данных, удаление данных из таблицы, удаление таблицы и др.);

– формирование и выполнение собственных SQL-запросов (в виде SQL-инструкции или в визуальном режиме с помощью конструктора запросов);

– создание копии базы данных;

– экспорт и импорт структуры и информации базы данных (сохранение во внешних файлах);

– создание пользователей и назначение полномочий.

Оболочка phpMyAdmin имеет интуитивно понятный графический интерфейс и не вызовет затруднения при наличии опыта работы с любой другой визуальной оболочкой СУБД (например, MS Access). Все действия с базой данных в оболочке phpMyAdmin сопровождаются выводом соответствующих SQL-инструкций, что облегчает дальнейшее написание кода PHP-сценария, осуществляющего взаимодействие с базой данных.

MySQL поддерживает транзакции (механизм, который позволяет интерпретировать множественные изменения в базе данных как единую операцию), откат транзакций, а также предоставляет множество функций для работы со строками, датой и временем.

Работа PHP-сценария с информацией из базы данных предполагает выполнение определенной последовательности действий:

1. Подключение к серверу баз данных.
2. Выбор используемой базы данных.
3. Формирование запроса к базе данных (SQL-инструкции).
4. Выполнение запроса.
5. Вывод полученных результатов запроса.
6. Разрыв соединения с базой данных.

Язык PHP имеет встроенные функции для работы с базами данных MySQL (функции специфичны именно для этой СУБД).

Подключение к серверу баз данных (функция возвращает дескриптор соединения): mysql_connect($адрес_SQL_сервера, $пользователь, $пароль).

Выбор базы данных для использования: mysql_select_db($имя_базы, $дескриптор_соединения).

Выполнение запроса к базе данных (функция возвращает указатель на данные, полученные с помощью запроса):

mysql_query($строка_SQL_инструкции).

Получение данных запроса (функция формирует массив с данными полученными в результате запроса mysql_query):

mysql_fetch_array($указатель_на_данные, формат),

формат – необязательный параметр, указывает тип массива: MYSQL_NUM – массив с нумерованными индексами (по умолчанию), MYSQL_ASSOC – массив с индексами по имени столбцов таблицы.

Разрыв соединения с базой данных:

mysql_close($дескриптор_соединения).

Поскольку выполнение запросов к базе данных происходит с помощью SQL-инструкций, приведем значение основных команд языка SQL (табл. 29).

Таблица 29. Основные команды SQL

Команда SQL	Назначение команды
CREATE DATABASE	Создание новой базы данных
SHOW DATABASES	Вывод списка существующих баз данных
USE DATABASE	Выбор базы данных
DROP DATABASE	Удаление базы данных
SHOW TABLES	Вывод списка существующих в базе данных таблиц

CREATE TABLE	Создание таблицы
DESCRIBE	Вывод характеристик полей таблицы
Команда SQL	Назначение команды
DROP TABLE	Удаление таблицы
INSERT INTO	Ввод данных в таблицу
SELECT	Вывод данных (запросы к базе данных, выборка)
UPDATE	Изменение данных, хранящихся в таблице
DELETE FROM	Удаление строк (записей) таблицы
ALTER TABLE	Операции со столбцами (полями) таблицы

Рассмотрим пример. Пусть в MySQL создана база данных data, хранящаяся на локальном сервере localhost. База данных data содержит одну таблицу users_data, предназначенную для хранения информации о пользователях (имя, пол, номер телефона). Структура таблицы users_data приведена на рис. 22.

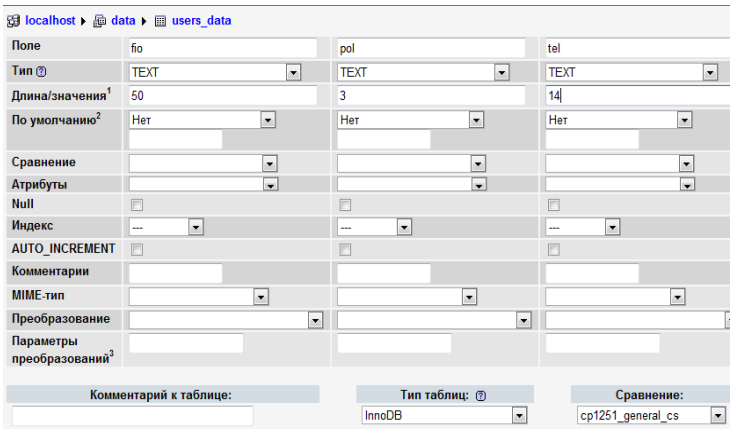
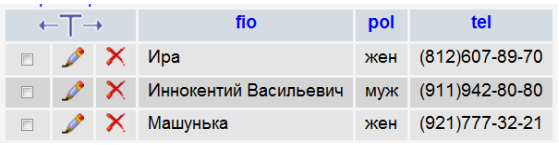


Рис. 22. Структура таблицы users_data

Таблица users_data содержит данные, представленные на рис. 23.



	fio	pol	tel
	Ира	жен	(812)607-89-70
	Иннокентий Васильевич	муж	(911)942-80-80
	Машунька	жен	(921)777-32-21

Рис. 23. Данные таблицы users_data

Сценарий db_connect.php осуществляет подключение к серверу и выбор базы данных:

```
<?php
// Информация о базе данных
$db_host="localhost";
$db_database="data";
$db_username="root";
$db_password="";
// Подключение к базе данных
$connection = mysql_connect($db_host, $db_username, $db_password);
// Указание кодировки соединения
mysql_query("set names cp1251");
// Выбор базы данных
$db_select = mysql_select_db($db_database);
?>
```

В процессе подключения или выбора базы данных могут возникнуть ошибки, связанные с неправильным указанием параметров базы данных либо отсутствием необходимых объектов. В этом случае дальнейшая работа с базой данных станет невозможной. Поэтому на период отладки следует добавить в сценарий подключения обработку ошибок, например:

```
<?php
// Информация о базе данных
$db_host="localhost";
$db_database="data";
$db_username="root";
$db_password="";
// Подключение к базе данных
$connection = mysql_connect($db_host, $db_username, $db_password);
//Обработка ошибок подключения
if (!$connection) {
echo("<p>Невозможно подключиться к базе данных:<br/>".
mysql_error()."</p>");
exit();
}
// Указание кодировки соединения
mysql_query("set names cp1251");
// Выбор базы данных
$db_select=mysql_select_db($db_database);
// Обработка ошибок выбора базы данных
if (!$db_select) {
echo("<p>Невозможно выбрать базу данных: <br />".
mysql_error()."</p>"); exit();
}
?>
```

Выдача информации о причине ошибки работы с базой данных MySQL производится функцией mysql_error().

Вместо пары функций echo() и exit() можно выдать сообщение об

ошибке непосредственно из функции exit() или die(), которая остановит исполнение сценария.

После того как соединение с базой данных установлено, можно приступить к исполнению SQL-запросов. SQL-инструкцию можно сформировать в phpMyAdmin, а затем скопировать. Сценарий show_info.php выводит все данные из таблицы users_data, а затем закрывает соединение с базой данных. Закрытие соединения освобождает все занимаемые им ресурсы и память. В сценарий добавлена проверка выполнения запроса к базе:

```
<?php
// Подключение к базе данных
include("db_connect.php");
// Запись в переменную текста запроса
$query="SELECT * FROM `users_data`";
// Выполнение запроса к базе данных
$result=mysql_query($query);
//Проверка выполнения запроса
if (!$result)
die("<p>Невозможно выполнить запрос: <br />" . mysql_error(). "</p>");
else
// Так как запрос выдает несколько строк,
//для вывода используется цикл
while($showdata=mysql_fetch_row($result, MYSQL_ASSOC))
echo("<p><b>" . $showdata["fio"] . "</b> ". $showdata["pol"] . " ". $show-
data["tel"]. "</p>");
//Разрыв соединения
if(!mysql_close($connection))
echo("<p>Невозможно разорвать соединение с базой данных</p>");
?>
```

Результат выполнения этого сценария приведен на рис. 24.

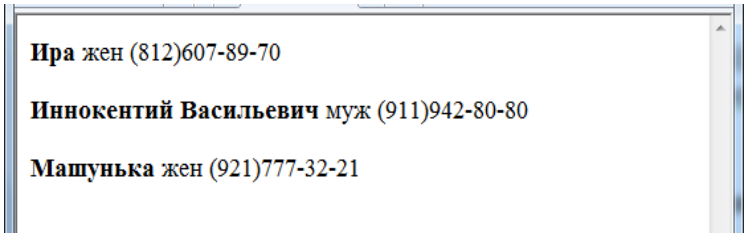


Рис. 24. Вывод PHP-сценарием информации из базы данных

Варианты вывода из базы данных (состав полей, порядок отображения данных, отбор по условию и т. п.) различаются лишь текстом SQL-инструкции.

Другая задача – занесение получаемых из web-формы пользовательских данных в базу. Пусть заполнение данных пользователем производится с помощью Web-Формы, поля формы имеют имена fio, pol и tel, данные формы передаются серверному сценарию с помощью метода post. Пусть также серверный сценарий, обрабатывающий данные формы (рассмотрен в предыдущем пункте, посвященном проверке данных Web-Форм), после выполнения всех необходимых проверок вызывает сценарий safe_info.php, который должен сохранить полученную информацию в таблице users_data.

Текст сценария safe_info.php может быть следующим:

```
<?php
// Подключение к базе данных
include("db_connect.php");
//Запись в переменную текста запроса на добавление записи
$sql="INSERT INTO`data`.`users_data`(`fio`,`pol`,`tel`) VALUES ('". $_POST["fio"]."', '".$_POST["pol"]."', '".$_POST["tel"]."');";
// Выполнение запроса к базе данных
$result=mysql_query($sql);
//Проверка выполнения запроса
if (!$result)
die("<p>Невозможно выполнить запрос: <br />". mysql_error()."</p>"); else
echo("<p>Данные успешно сохранены</p>");
//Разрыв соединения
if(!mysql_close($connection))
echo("<p>Невозможно разорвать соединение с базой данных</p>");
?>
```

Работа с базами данных может осуществляться с помощью расширения PDO (PHP Data Objects) - библиотеки классов, предоставляющей универсальный интерфейс доступа к различным базам данных. PDO входит в состав PHP, начиная с версии 5.1, в более ранних версиях PDO не работает.

PDO позволяет использовать унифицированные методы для работы с различными базами данных, хотя текст запросов может немного отличаться, так как многие СУБД реализуют свой диалект SQL.

PDO не использует абстрактных слоев для подключения к базам данных, а применяет драйверы самих баз данных (рис. 25), что позволяет добиться высокой производительности.

Рис. 25. Взаимодействие PHP-сценария с базой данных с помощью расширения PDO

Расширение поддерживает любую базу данных, для которой есть PDO-драйвер. В настоящее время для PDO существуют драйверы практически ко всем общеизвестным СУБД и интерфейсам (MySQL, MS SQL Server, PostgreSQL, ODBC, Oracle Call Interface и др.).

Просмотр списка доступных в системе драйверов осуществляется командой print_r(PDO::getAvailableDrivers());.

Универсальные функции PDO используют ту же основную информацию, что и PHP-функции работы с MySQL, но при вызове дополнительно указывается тип базы данных, с которой осуществляется взаимодействие. Далее примеры команд приведены для СУБД MySQL.

Подключение к базе данных с помощью PDO имеет следующий вид:

```
$connection=new PDO("mysql:host=$db_host;dbname=$db_database", $db_username, $db_password);
```

Рекомендуется использовать механизм исключений из библиотеки PDO (PDOException), чтобы в случае неудачного подключения к базе данных получить сообщение об ошибке. Пример обработки исключений с помощью конструкции try…catch:

```
try {
$connection = new PDO("mysql:host=$db_host;dbname=$db_database", $db_username,
$db_password);
//Установка режима обработки ошибок
$connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
catch(PDOException $e) { echo $e->getMessage();
}
```

В режиме PDO::ERRMODE_EXCEPTION необработанные ошибки будут вызывать исключения и остановку выполнения сценария. В этом примере используются операции работы с объектами: new – создание нового экземпляра объекта, -> – вызов свойства или метода объекта.

Сценарий db_connect.php для подключения базы данных data с использованием PDO примет следующий вид:

```
<?php
// Информация о базе данных
$db_host="localhost";
$db_database="data";
$db_username="root";
$db_password="";
// Подключение к базе данных
try {
$connection = new PDO("mysql:host=$db_host;dbname=$db_database",
$db_username,$db_password);
//Установка режима обработки ошибок
$connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
// Задание кодировки
$connection->query("SET NAMES cp1251");
```

```
}
//Обработка исключений catch(PDOException $e) { die($e->getMessage());
}
?>
```

Для исполнения SQL-инструкции используется метод query()

дескриптора соединения. Пример:

```
$query="SELECT * FROM `users_data`";
$result=$connection->query($query).
```

Для вывода информации, полученной из базы данных, используется метод fetch() дескриптора выполнения запроса (указателя на данные). Перед использованием метода можно указать, в каком виде будут возвращены данные, например: PDO::FETCH_ASSOC – массив, индексированный по именам столбцов, PDO::FETCH_BOTH – массив, индексированный по именам столбцов и по номерам (по умолчанию), PDO::FETCH_NUM – массив, индексированный по номерам столбцов, и т. д.

Пример установки метода извлечения данных:

```
$result->setFetchMode(PDO::FETCH_ASSOC);
```

Пример вывода данных (используется формат вывода, установленный по умолчанию):

```
while($showdata=$result->fetch()) echo("<p><b>".$showdata["fio"]."</b> ".$showdata["pol"]." ". $showdata["tel"]."</p>");
```

Разрыв соединения производится назначением дескриптору соединения значения NULL, например: $connection = null;.

Сценарий show_info.php для вывода информации из базы данных с использованием PDO примет вид:

```
<?php
// Подключение к базе данных
include("db_connect.php");
// Запись текста запроса в переменную
$query="SELECT * FROM `users_data`"; try {
// Выполнение запроса к базе данных
$result=$connection->query($query);
// Так как запрос выдает несколько строк, используется цикл while($showdata=$result->fetch()) echo("<p><b>".$showdata["fio"]."</b> ".$showdata["pol"]. " ". $show- data["tel"]."</p>");
// Разрыв соединения
$connection=null;
}
//Обработка исключений catch(PDOException $e) { die($e->getMessage());
}
?>
```

Вставка новых данных (INSERT INTO) или обновление существующих (UPDATE) - наиболее часто используемые общие операции баз данных.

При использовании PDO для реализации этих операций можно применять метод exec(), например:

$sql = "INSERT INTO `data`.`users_data` (`fio`, `pol`, `tel`) VALUES ('".$_POST["fio"]."', '".$_POST["pol"]."', '" . $_POST["tel"]."');";

$result=$connection->exec($sql);

Однако в целях безопасности рекомендуется использовать подготовленные выражения, что разбивает каждую из операций (изменение, вставка данных) на два этапа (рис. 26) с применением методов prepare() и execute().

Рис. 26. Схема выполнения обновления и добавления данных с использованием подготовленных выражений

Подготовленные выражения – это предварительно скомпилированные инструкции SQL, которые могут быть выполнены много раз с пересылкой на сервер только данных. Подготовленные выражения имеют большую скорость выполнения, а также позволяют четко разделить структуру и входные данные запроса. Кроме того, их использование обеспечивает защиту от атак типа SQL-инъекция, так как все передаваемые в них параметры автоматически экранируются.

Если подготовленные выражения не используются для экранирования специальных символов в пользовательских данных, можно использовать метод дескриптора соединения quote(), например:

$fio=$connection->quote($_POST["fio"]);.

Работа с подготовленными выражениями выполняется следующим образом. Сначала следует сформировать шаблон инструкции SQL, который будет затем скомпилирован.

Пример *неименованного* шаблона:

$sql = "INSERT INTO `data`.`users_data` (`fio`, `pol`, `tel`) VALUES (?, ?, ?);";

Пример *именованного* шаблона:

$sql = "INSERT INTO `data`.`users_data` (`fio`, `pol`, `tel`) VALUES (:fio, :pol, :tel);";

Подготовка выражения осуществляется с помощью метода prepare() дескриптора соединения, например: $result = $connection->prepare($sql);.

Затем следует определить данные, которые будут подставлены в шаблон. Легче всего это сделать через объявление массива. Выполнение шаблона запроса, заполненного данными, осуществляется методом execute().

Пример объединения с данными и выполнения запроса с

неименованным шаблоном:

```
$data = array($_POST["fio"], $_POST["pol"], $_POST["tel"]);
$result->execute($data);
```

Пример объединения с данными и выполнения запроса с именованным шаблоном:

```
$data = array("fio" => $_POST["fio"], "pol" => $_POST["pol"], "tel" =>
$_POST["tel"]);
$result->execute($data);
```

Остановимся на варианте использования неименованного шаблона. Тогда сценарий safe_info.php, выполняющий сохранение пользовательского ввода в базе данных, с использованием PDO и подготовленных выражений примет вид:

```
<?php
// Подключение к базе данных
include("db_connect.php");
// Запись в переменную шаблона запроса
$sql = "INSERT INTO `data`.`users_data` (`fio`, `pol`, `tel`) VALUES (?, ?,
?);";
try {
// Подготовка выражения
$result = $connection->prepare($sql);
// Подготовка данных для вставки в шаблон
$data = array($_POST["fio"], $_POST["pol"], $_POST["tel"]);
// Выполнение запроса к базе данных
$result->execute($data);
echo ("Данные успешно сохранены");
// Разрыв соединения
$connection=null;
}
//Обработка исключений catch(PDOException $e) { die($e->getMessage());
}
?>
```

Итак, были подробно рассмотрены методы работы с базами данных, как специфичные для СУБД MySQL, так и универсальные, позволяющие организовать взаимодействие с различными базами данных.

Вместе с тем, за пределами данного учебного пособия остались такие возможности языка PHP, как управление внешними файлами, загрузка файлов на сервер, работа с маркерами cookie, авторизация пользователей и управление сеансами (сессиями), а также вопросы обеспечения безопасности, связанные с решением этих задач.

Однако основы языка PHP в рассмотренном объеме позволяют понять наиболее важные концепции и принципы серверного программирования.

2.6. Знакомство с JavaScript

JavaScript – объектно-ориентированный сценарный язык программирования.

Области применения:

– динамическое создание Web-страницы;

– проверка достоверности полей формы до передачи на сервер;

– вывод сообщений для пользователей;

– учет особенностей браузера и монитора пользователя для корректного отображения;

– составной элемент технологии AJAX;

– счетчики посещаемости.

Сценарий JavaScript встраивается в HTML-документ с помощью тега **<script>**. При этом скрипт может располагаться в произвольном месте страницы.

Пример. Выводит слово «Привет» в теле документа.

```
<html>
<head>
</head>
<body>
<script> document.write("Привет!");
</script>
</body>
</html>
```

Здесь объект document – это HTML-документ, загруженный в окно браузера. Метод write записывает в тело HTML-документа строку "Привет!".

Переменные

Можно объявлять с помощью ключевого слова var, а можно использовать без объявления.

```
var th=12
age=12
var dd=‖qwe‖
```

Операторы языка JavaScript

В данном подразделе рассматриваются основные операторы языка JavaScript. При этом полный список операторов не ограничивается приведенными.

Операторы присваивания

	Присваивание
=	Сложение или слияние строк (x=x+2 аналогично x+=2)
=	Вычитание (x=x-3; аналогично x-=3)

=	Умножение
=	Деление

Операторы сравнения

>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
==	Равно
!=	Не равно

Унарные операторы

++	Увеличение значения переменной на единицу. Может применяться как префикс и как суффикс
--	Уменьшение значения переменной. Может применяться как префикс и как суффикс

Чтобы увидеть разницу между использованием унарных операций как префикс и суффикс, изучите работу двух кодов:

```
a=1                          a=1
document.write(a++)          document.write(++a)
```

В первом случае будет выведено значение 1, во втором – 2.

Условные операторы

if-else

Пример. Поиск максимального значения из двух чисел if(a<b) {max=b} else {max=b}

Последний пример можно реализовать при помощи такой более компактной конструкции:

max=(a<b) ?b:a

switch - case

Определяет действия в зависимости от значения переменной switch(variable) {

```
case value_1: {
//блок операторов_1 break;
}
case value_2: {
//блок операторов_2 break;
}
case value_n: {
//блок операторов_n break;
```

```
}
default: {
//блок операторов по умолчанию
}
}
```

Операторы цикла

for

Пример вывода пирамидки из горизонтальных линий.

Для этого достаточно набрать код: for(i=1;i<=10;i++) document.write("<hr width=\""+i*10+"%\">")/

Таким образом, первый аргумент – это начальное значение, второй – конечное, третий – изменение. В примере использована конструкция вида \", которая называется экранирование. Цель – поместить внутрь строковой переменной кавычки для вывода на экран.

FOR-IN позволяет получить полный список свойств, методов, событий определенного объекта. Например.

```
for (props in document)
document.write(props+"<br>");
```

WHILE

Пример. Вывести таблицу умножения для введенного числа. a=prompt("Введите число",6)

```
i=1
while(i<=10)
{document.write(a+"x"+i+"="+a*i+"<br>") i++}
```

Кроме этих операторов в организации цикла могут участвовать еще два оператора: **break** (выход из цикла) и **continue** (переход на следующий шаг).

Функции

Функции определяются так: function F_name(аргументы)

{тело функции}

Замечания

1. Операторы JavaScript напоминают общеизвестные операторы языка C++.
2. После оператора надо ставить точку с запятой, но если в строке используется только одна команда, то точку с запятой можно не ставить.
3. Для объединения группы операторов используются фигурные скобки.
4. Однострочные комментарии задаются при помощи двух слешей //
5. Многострочные комментарии при помощи конструкции /* в начале и */ в конце.

Объекты JavaScript

Объект ***Array***

Для создания массива можно воспользоваться одним из трех способов:

```
abc=new Array()
abc=new Array(10)
abc=new Array("Синий","Красный","Зеленый")
```

Объект Array имеет одно свойство **length** -– число элементов массива. Методы объекта Array:

– join( ). Слияние элементов массива в строку. Через параметр передается разделитель элементов. По умолчанию разделителем служит запятая.

– reverse( ). Меняет порядок элементов массива на обратный. Пример. Создание аналога gif-рисунков.

```
<html>
<head>
<title>Создаем анимацию
</title>
<script> i=0
function ShowNextImage()
{
iii.src=i + ".jpg" i++
setTimeout ("ShowNextImage()",1000) if ((i>=4)) i=0
}
</script>
</head>
<body>
<img src="0.jpg" name="iii">
<script> ShowNextImage()
</script>
</body>
</html>
```

В данном примере заранее заготовлены рисунки с названиями 0.jpg … 4.jpg. Если бы мы хотели использовать произвольные названия, то вместо команды iii.src=i + ".jpg" следовало бы использовать команду iii.src=fname[i], где fname – массив, содержащий названия файлов.

Объект ***Date***

Предоставляет функции для работы с датой и временем.

Создание объекта newDate= new Date()

Методы:

getDate() – возвращает число месяца как целое число;

getDay() – возвращает день недели: getHours() getMinutes() getSeconds() getYear() getMonth();

getTime() – возвращает количество миллисекунд прошедших с 0:0:0 1 января 1970 года.

Если вместо **get** указать **set**, то метод устанавливает соответствующее

значение в переменной. Например, newDate.getYear(2015) установит значение года, равное 2015.

Следующий пример при обновлении страницы выводит число секунд, которые пользователь был на сайте.

```
<html>
<script>
function person_in()
{
t1=new Date()
}
function person_out()
{
t2=new Date() t3=t2.getTime()-t1.getTime() t3=Math.round(t3/1000)
alert("Вы были на сайте " + t3+ "c")
}
</script>
<body onLoad="person_in()" onUnLoad="person_out()">
</body>
</html>
```

В данном примере использованы две написанные функции:

– person_in(), которая вызывается при открытии страницы, – обработчик onLoad записан в теге body. Данная функция сохраняет значение времени в переменной t1;

– person_out(), которая вызывается при открытии страницы, – обработчик onUnLoad. В настоящее время большинство браузеров не поддерживает этот обработчик.

Объект ***Math***

Предназначен для работы с математическими константами и функциями. Приведем некоторые свойства и методы.

Свойства: E, PI, SQRT, SQRT1_2.

Методы: cos(), sin(), exp(), abs() - модуль, round() - округление, ceil() - округление вверх, floor() - округление вниз, pow(число1, число2) - возведение в степень, sqrt() - корень квадратный.

Объект ***String***

Предназначен для работы со строковыми значениями. Создать переменную можно двумя способами: MyString=‖значение‖

MyString=new String(‖значение‖)

Одно свойство – length – длина переменной.

Методы (частично):

big() – возвращает строку с добавлением тегов <big>…</big>; bold() - возвращает строку с добавлением тегов …;

charAt(позиция) – возвращает символ, стоящий на указанной позиции;

fontcolor(‖цвет‖) – изменяет цвет;

sub(), sup() – индексы;

substring(позиция1, позиция2) – возвращает подстроку, начинающуюся символом в первой позиции и заканчивающуюся перед второй позицией;

toLowerCase() – преобразует исходную строку в строку со строчными символами;

toUpperCase() – преобразует исходную строку в строку с заглавными символами.

Пример. В текстовом элементе формы по надписи пробегает заглавная буква.

```
<form name="f1">
<input type="text" name="t1" size="100">
</form>
<script>
	str="Сыктывкарский государственный университет им. Питирима Сорокина"
i=0
function qwe()
{
i++
document.f1.t1.value=str.substring(0,i)  +  str.charAt(i).toUpperCase()
+ str.substring(i+1, str.length)
setTimeout("qwe()",300)
}
qwe()
</script>
```

Некоторые встроенные функции JavaScript:

– isNaN(значение) - возвращает истину, если значение не является числом;

– parseFloat(строка) - преобразует строку в число с плавающей точкой;

– parseInt(строка,основание) - преобразует строку в целое число по указан ному основанию;

– typeOf(объект) - возвращает тип указанного объекта как строку.

Формы HTML. Обработка форм с помощью JavaScript

Основные элементы формы

Форма - это элемент HTML, позволяющий передавать информацию на Web-сервер, где информация будет обработана. С помощью форм организуются тесты, голосования, опросы. Заметим, что html-формы сами по себе позволяют только организовывать ввод информацию. Для обработки форм необходимо использовать языки программирования, для обработки на стороне клиента можно использовать, например, язык JavaScript, а на стороне сервера - например, PHP, Perl, C.

Форма задается тегами **<form></form>.** Все остальные элементы формы располагаются между этими тегами. Тег <form> может иметь несколько параметров:

– name – имя формы. Необходимо, если на странице несколько форм$

– action – определяет URL-адрес, по которому будет отправлена информация введеная пользователем$

– method – определяет способ отправки информации.

– target – указывает имя окна, в котором будут отображаться результаты обработки отправленной формы.

Рассмотрим элементы, располагающиеся в форме. Начнем с *текстового поля*. Оно задается тегом <**input**>.

Пример использования

Исходный код	Вид в браузере
<form name="forma1"> <input type="text" name="t1" size="20" maxlength="50" value="Введите текст"> </form>	Введите текст

Параметры тестового поля:

– name – имя элемента;

– type – тип элемента (в данном случае - text);

– size – размер текстового поля в символах, которые одновременно будут видны, при вводе большего количества символов, они будут прокручиваться;

– maxlength – максимальное количество символов, которое можно ввести в поле, если опустить этот параметр, то число символов будет неограниченным;

– value - текст, который будет отображаться (его можно стереть), при отсут- ствии этого параметра поле будет пустым;

– disabled - блокирует поле от любых изменений;

– readonly - делает поле доступным только для чтения.

Текстовое поле для ввода пароля. Представляет собой текстовое поле с тем отличием, что вводимый текст не отображается, вместо него появляются специальные символы, например звездочки. Используется при вводе паролей. От текстового поля отличается только параметром type="password".

Флажки. Начнем с примера. Ваши любимые дисциплины:

☐ Философия ☐ Психология ☐ Математический анализ

Флажки задаются тегом <input>, причем один тег задает один флажок. Поэтому чтобы задать три флажка, надо три раза писать input. Для задания указанного выбора необходимо задать следующий код:

<form name="forma1">

Ваши любимые предметы:

<input type="checkbox" checked> Философия

<input type="checkbox" > Психология

<input type="checkbox" > Математический анализ
</form>.

Рассмотрим параметры тега <input>:

– type - тип элемента (в данном случае - checkbox),

– name - имя элемента, указывает программе обработчику формы, какой пункт выбрал пользователь,

– value - значение элемента, указывает программе обработчику формы значение пункта, который выбрал пользователь,

– checked - им обычно помечают наиболее вероятные для выбора пункты, пользователь щелчком мыши может выбрать другие пункты.

Переключатели. В отличие от флажков здесь можно выбрать только один пункт. В связи с этим значения параметра name должны быть одинаковы для всех элементов группы. Параметр type="radio", все остальные такие же, как у флажков.

Кнопки. Существует четыре типа кнопок:

– submit - кнопка отправки содержимого формы Web-серверу. Ее параметры:

- type="submit" - тип кнопки,
- name - имя кнопки,
- value - надпись на кнопке;

– image - графическая кнопка отправки содержимого формы web-серверу. Для ее использования необходимо подготовить картинку кнопки, а потом использовать ее в виде кнопки. Ее параметры:

- type="image" - тип графической кнопки,
- name - имя кнопки,
- src - адрес картинки для кнопки;

– reset - кнопка, позволяющая восстановить все значения по умолчанию в форме. Ее параметры:

- type="reset" - тип кнопки очищения,
- name - имя кнопки,
- value - надпись на кнопке;

– button - произвольная кнопка, ее действия назначаются вами с использованием языков программирования, т.е. сама она делать ничего не умеет. Ее параметры:

- type="button" - тип произвольной кнопки,
- name - имя кнопки,
- value - надпись на кнопке.
- onclick - обработчик события - указывает, что делать при щелчке по кнопке. Вообще, у этого типа кнопок есть и другие события (например, двойной щелчок), но здесь мы не будем их рассматривать.

Кнопки можно задавать и при помощи тегов <**button**> </**button**>. Возможностейу таких кнопок несколько, они могут иметь содержимое в виде текста или картинки. Этот тег имеет следующие параметры:

– type - тип кнопки, может принимать значения:
  - reset - кнопка очистки формы,
  - submit - кнопка отправки данных;
  - button - кнопка произвольного действия.
– name - имя кнопки,
– value - надпись на кнопке.

Поле для файлов. Поле для ввода имени файла, сопровождаемое кнопкой Browse (Обзор), при щелчке по которой открывается окно просмотра дерева папок компьютера, где можно выбрать нужный файл. Выбранный файл присоединяется к содержимому формы при отправке на сервер.

Пример.

```
<form name="forma1">
<input type="file" name="load" size="50">
</form>
```

Поле для ввода текста. Для больших текстов, например для почтовых сообщений, удобно использовать именно этот элемент. Он создается тегами <**textarea**> </**textarea**> и имеет следующие параметры:

– name - имя поля,
– cols - ширина поля в символах,
– rows - количество строк текста, видимых на экране,
– wrap - способ переноса слов:
  - off - переноса не происходит,
  - virtual - перенос отображается, но на сервер поступает неделимая строка,
  - physical - перенос и на экране и при поступлении на сервер;
– disabled - неактивное поле;
– readonly - разрешено только чтение.

Раскрывающиеся списки. Списки бывают с возможностью выбора одного элемента и с множественным выбором. Задются и те и другие с помощью тегов <**select**> </**select**>, внутри которых располагаются элементы значений, заданных тегом <**option**>.

Параметры этих тегов.

<select>:

– name – имя списка. Каждый выбранный элемент списка при передаче на сервер будет иметь вид: name.value, где значение (value) берется из тега option;

– size – определяет количество видимых элементов в списке: 1 – простой раскрывающийся список, больше 1– список с полосой прокрутки;

– multiple - разрешает выбор нескольких элементов списка.

<option>:

– selected – им помечают наиболее вероятный для выбора элемент списка, если список со множественным выбором, то можно пометить

несколько пунктов;

– value – значение, которое будет отправлено серверу, если пункт выбран.

Можно использовать теги <**optgroup**> </**optgroup**>, позволяющие группировать элементы списка по каким-либо признакам.

Надписи. Все элементы формы можно связать с их надписями при помощи элемента <**label**> и его параметра for, значением которого является имя элемента, с которым связываем надпись. Например:

```
<form name="forma1">
<label for="load">Выбирайте файл: </label>
<input type="file" name="load" size="30">
</form>
```

Обращение к элементам формы

Обращение к значениям текстового поля и текстовой области

Первый способ – по именам элементов формы: document.fname.tname.value

Здесь fname – название формы, ftext – название поля или области. Кроме этого, все формы на странице образуют коллекцию, в свою очередь все элементы формы образуют также коллекцию. Поэтому доступно такое обращение:

document.forms[0].elements[2].value

Проверка checkbox

if (document.fname.check1.checked) {операторы}

Выпадающий список

Сначала определяется выбранный элемент, затем определяется значение:

```
idx=document.fname.sel1.selectedIndex
a=document.fname.sel1.options[idx].text
```

Обработчики событий

Приведем события, которые можно определить объектам формы и документа.

– **onBlur** происходит, когда поля формы или окно документа теряет фокус.

– **onFocus -** приобретение фокуса.

– **onChange -** изменение значения select.

– **onClick** - щелчок мыши.

– **onMouseOver** - при наведении мыши на объект.

– **onMouseOut** - при отведении мыши с объекта.

– **onLoad** - при загрузке документа.

– **onUnload** - при закрытии страницы.

– **onSelect** - при выделении текста.

– **onSubmit** - при нажатии кнопки Принять.

Объектная модель Dynamic HTML

Динамический HTML (DHTML) - совокупность приемов, позволяющих динамически изменять оформление и содержание Web-страницы в ответ на действия пользователя. DHTML является продуктом взаимодействия трех технологий: языка HTML, каскадных таблиц стилей (CSS) и языка сценариев (JavaScript). Для предоставления сценариям доступа к HTML и CSS содержимое документа представляется в виде дерева объектов в программной модели.

Window

Объект window – вершина иерархии объектной модели. Все остальные объекты являются дочерними или более отдаленными потомками объекта window. Браузер создает, как правило, единственный объект window, когда открывает документ в окне, однако если документ содержит фреймы (элементы frame и iframe), то дочерние объекты window создаются также для каждого фрейма. Доступ к дочерним окнам возможен через коллекцию frames родительского окна.

Полный список свойств, методов, событий объекта window (как и у других объектов) можно получить так:

```
<script>
sprops="<H2>Свойства объекта window</H2>" for(props in window)
    sprops+="<b>"+props+"</b><br>"; document.write(sprops);
</script>
```

Свойства объекта window:

– parent;
– self;
– top;
– document;
– event;
– history;
– location;
– navigator;
– screen.

Методы:

– alert - вывод сообщения;
– prompt - ввод данных;
– confirm - подтверждение;
– close - закрыть окно;
– open - открыть окно;
– blur - убрать фокус;
– focus - установить фокус;

– navigate - загрузить страницу;
– scroll - развернуть страницу на указанный размер;
– setTimeout() - загрузить функцию через указанное время;
– cleartimeout() - удалить загрузчик функции;
– showModalDialog - открыть модальное окно.

Document

Рассмотрим некоторые свойства, коллекции и методы документа, загруженного в окно браузера.

Свойства:

– linkColor - цвет ссылок;
– fgColor - цвет щрифта;
– bgColor - цвет фона. Коллекции
– all - все теги на странице;
– anchors - все ссылки;
– forms - коллекция форм.

Методы:

– clear - очистить документ;
– close - закрыть;
– open - открыть;
– write - записать в документ.

Объект navigator

Объект navigator содержит информацию о браузере.

Свойства:

– userAgent содержит строку, которая передается в HTTP заголовке на веб- сервер с каждым запросом. Эта строка содержит информацию о самом браузере и его версии, операционной системе и ее версии и некоторую дополнительную информацию;
– appName содержит имя;
– appCodeName – «Mozilla» для большинства браузеровappVersion, appMinorVersion, platform, systemLanguage и другие содержат информацию о версии браузера и операционной системы, языковые на- стройки и другое.

Объект location

Объект location содержит полную информацию об адресе страницы.

Свойства:

– href содержит всю строку URL;
– protocol - протокол передачи данных;
– host - имя сервера + порт (www.ya.ru:80);
– hostname - имя сервера (www.ya.ru);
– port - порт;
– pathname - путь;
– search - строка запроса (?a=html&page=1);

– hash - закладка, ссылка внутри страницы (#label1).

Объект history

Объект history содержит информацию об адресах, посещенных пользователем с момента открытия браузера.

Методы:

– back() - переход на предыдущую страницу. Эквивалентно нажатию кнопки «Назад» в браузере.

– forward() - переход на следующую страницу. Эквивалентно нажатию кнопки «Вперед» в браузере.

– go(n) - переход на n позиций в индексе

Объект screen

Объект screen содержит информацию о мониторе: ширина - width, высота - height, глубина цвета - colorDepth.

Объект event

Событие - это сообщение, отправляемое в ответ на некоторое действие, например завершение загрузки документа, перемещение или щелчок мыши, нажатие клавиши. Обработчик события - это, как правило, функция, написанная на языке сценариев (например, JavaScript), которая выполняет действия, когда произошло соответствующее событие.

Объект window.event создается браузером автоматически в момент возникновения. Следует подчеркнуть, что модель объекта event в Internet Explorer существенно отличается от используемой в других браузерах. Тем не менее целый ряд свойств оказывается одинаковым во всех современных браузерах.

– type - тип события без приставки "on". Например, событие onclick имеет тип click.

– button - нажатая кнопка мыши (0 - ни одна, 1 - левая, 2 - правая, 3 - левая и правая, 4 - средняя, 5 - левая и средняя, 6 - правая и средняя, 7 - все три). Это свойство имеет смысл для событий мыши. В случае остальных событий имеет значение 0 независимо от действительного состояния кнопок.

– clientX, clientY - координаты указателя мыши относительно верхнего левого угла клиентской области окна браузера.

– screenX, screenY - координаты указателя мыши относительно верхнего левого угла дисплея.

– keyCode - целочисленный код клавиши, которая вызвала событие клавиатуры. Коды клавиш различаются для различных аппаратных платформ (PC или Macintosh).

– shiftKey, ctrlKey, altKey - истина (true), если во время события нажата также клавиша Shift, Ctrl или Alt соответственно.

– cancelBubble - запрещает (true) или разрешает (false, по умолчанию) передачу события вверх по иерархии от элемента-источника к родительскому элементу. Каждое событие происходит на единственном элементе-источнике. Если элементу-источнику не сопоставлена функция-обработчик, событие

передается вверх по дереву элементов, пока не встретит функцию-обработчик или не достигнет корневого элемента. Если функция-обработчик найдена, но не содержит cancelBubble=true, то передача вверх продолжается, если содержит, то прерывается.

Отметим некоторые свойства, доступные в Internet Explorer:

– элемент-источник события доступен в IE как свойство srcElement, в Mozilla Firefox и других как target^;

– события onmouseover и onmouseout в Internet Explorer показывают свойства-объекты откуда (fromElement) и куда (toElement). В Mozilla Firefox эти объекты являются свойством relatedTarget.

Обращение к свойствам таблицы стилей

Если известен идентификатор объекта (например, id1) и необходимо обратиться к свойству left, то можно обратиться так:

id1.style.left=‖100 px‖

Свойства и методы управления содержимым

Свойства:

– innerText - заменяет текст, содержащийся в элементе;

– outerText - заменяет весь элемент;

– innerHTML - заменяет текст и код элемента;

– outerHTML заменяет весь текст и код.

Пример. Имеется заголовок третьего уровня

<h3 id="heading1">Старый заголовок</h3> Далее в скрипте JavaScript даем командуы objHead1=document.all["Heading1"] objHead1.innerText="Новый заголовок " Методы:

– insertAdjacentText(положение, текст) - вставляет только текст;

– insertAdjacentHTML(положение, текст) - вставляет текст и HTML. Варианты положения: BeforeBegin, AfterBegin, BeforeEnd, AfterEnd. Пример. objHead1.incertAdjacentText(–AfterBegin‖, –Привет‖).

Пример реализации раскрывающихся списков с использованием приведенных свойств и методов.

```
<html>
<head>
<title>
Раскрывающийся список сиспользованием Inner
</title>
<script> i1t=0 i2t=0 i3t=0
function qwe()
{
Obj=window.event.srcElement if(Obj.id.length==2)
{ if(eval(Obj.id+"t==0"))
{

Obj.insertAdjacentHTML("BeforeEnd",document.all[Obj.id+"1"].innerHTML)
```

```
eval(Obj.id+"t=1") }
else
{
Obj.children[0].outerHTML="" eval(Obj.id+"t=0")}
}
}
</script>
</head>
Раскрывающийся список с использованием Inner
<div id=i11 style="display:none"><ul><li>1.1<li>1.2<ul>          </div>
<div id=i21 style="display:none"><ul><li>2.1<li>2.2<ul>          </div>
<div id=i31 style="display:none"><ul><li>3.1<li>3.2<ul>          </div>
<ul onClick="qwe()">
<li id=i1>1
<li id=i2>2
<li id=i3>3
</ul>
</html>
```

Модель DOM

Структура документа DOM

DOM - Document Object Model - программный интерфейс XML (и HTML) документов.

Уровни DOM

– Уровень 0. Включает в себя все специфические модели DOM. Необходимо обратить внимание, что эти модели формально являются не спецификациями DOM, опубликованными W3C, а, скорее, информацией о том, что существовало до начала процесса стандартизации.

– Уровень 1 (1998). Появление понятия узел.

– Уровень 2 (2000-2004). Поддержка пространства имѐн.

– Уровень 3 (с 2004).

DOM представляет документ как иерархию узлов (node). На вершине иерархии узел - document, который представляет собой весь документ. В качестве узлов представлено все содержимое документа: HTML-элементы, атрибуты и текст.

Для обсуждения свойств и методов будем использовать следующий фрагмент:

```
<div>
<ul id="components">
<li>HTML</li>
<li>CSS</li>
<li>JavaScript </li>
</ul>
</div>
```

Навигация по дереву документа

Навигацию можно начать с любого узла, у которого известен идентификатор.

var oList=document.getElementById("components")

Также в случае корректной страницы можно указать document.documentElement - самый верхний уровень.

Ссылка на родительский элемент: var oParent=oList.parentNode

Дочерние элементы представляют собой коллекцию, на элемент которой ссылаются так:

var oItem1=oList.childNodes.

В данном случае это будет элемент CSS.

На первый и последний дочерний элемент ссылаются так: firstChild, lastChild.

Кроме этого, можно сослаться на соседние узлы одного уровня:

var oPreviousSibling=oList.previousSibling

var oNextSibling=oList.nextSibling.

Создание новых узлов

Методы: createElement(), createTextNode().

Рассмотрим, как добавить к нашему списку элемент <li> XML.

Первым делом создаем узел LI. Для этого даем команду var oItem=document.createElement("LI")

Рекомендуется писать название тегов заглавными буквами.

Далее создаем содержимое узла:

var oText=document.createTextNode("XML")

Обращаем внимание на то, что узлы созданы, но еще не связаны ни между собой, ни с основным списком.

Редактирование дерева документов

Приведем некоторые методы для редактирования дерева документов. Вставка: appendChild(), insertBefore().

Копирование: cloneNode(). Замещение: replaceChild(). Удаление: removeChild().

Присоединим узел oText к узлу oItem:

oItem.appendChild(oText)

Теперь в памяти имеется веточка XML. Добавим ее в конец узла oList: oList.appendChild(oItem)

После последней команды в списке должен появиться элемент XML. Для вставки в произвольное место можно использовать метод insertBefore(), например, так: oList.firstChild.nextSibling.insertBefore(oItem)

Для копирования узла используется метод cloneNode(). Пример: var oClone=oList.cloneNode(true)

Если установлен параметр true, то узел копируется со всеми потомками.

Работа с массивами элементов

Свойство getElementById позволяет обращаться к элементу, у которого известен идентификатор. В случае, когда надо работать с определенными тегами, полезным является свойство getElementsByTagName, которое

возвращает все указанные теги в виде массива.

Пример. Выделение красным цветом каждой второй строки таблицы с использованием метода getElementsByTagName.

```
<body>
<table border="1">
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</body>
<script>
a=document.getElementsByTagName("tr") n=a.length
for(i=0;i<n;i=i+2)
{a[i].style.color="red"}
</script>
```

Библиотека jQuery

Введение в jQuery

jQuery - библиотека JavaScript, которая позволяет получать доступ к любому элементу модели DOM, обращаться к атрибутам и содержимому, а также манипулировать ими. Кроме этого, библиотека предоставляет удобный API для работы с технологией AJAX.

Библиотека была создана в 2006 г. В настоящее время практически каждый сайт использует данную библиотеку.

Подключив библиотеку jQuery вместо десятков команд на JavaScript можно написать несколько команд. Команды основаны на селекторах и классах CSS.

Библиотеку jQuery можно скачать с сайта http://jquery.com/. Размер библиотеки в минимальном варианте составляет примерно 60 Кб.

Перед началом работы библиотеку надо подключить:

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

Вся работа с библиотекой ведется с использованием функции $. Общая идея использования: 1) выбирается элемент или группа элементов, 2) выполняются действия над выделенными элементами.

Пример. Выделение красным цветом каждой второй строки таблицы.

```
<script src="jquery.js" type="text/javascript"></script>
<style>
.rrr{color:red}
</style>
<body>
<table border="1">
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
```

```
<script>
$("table tr:even").addClass("rrr")
</script>
```

Обращение к элементам

Рассмотрим на примерах варианты обращения. Обращаем внимание, что все основано на синтаксисе CSS.

1. $(p a) выбирает все ссылки, расположенные в абзацах.
2. $(' #myId ') - выбор элемента с указанным идентификатором.
3. $('myClass') - использование классов.
4. $('body>div') - выбор элементов div, являющихся прямыми потомками элемента body.
5. $(p:even) - четные абзацы.
6. Использование ссылок:

– a[href^=http://] - этому селектору соответствуют все ссылки, которые начинаются с символов http://, на это указывает символ ^/;

– a[href$=.pdf] - ссылки заканчиваются на «pdf»;

– a[href*=ya.ru] - ссылка содержит упоминание ya.ru в произвольном месте.

7. Имеется возможность выбора определенных элементов при условии, что они содержат другие элементы (в примере ниже двоеточие как в математике означает ‒такое что‖). Например, селектор li:has(a) выбирает элементы <li>, которые содержат элемент <a>.

8. Выбор по позиции (некоторые варианты):

– :first - первое совпадение на странице;

– :last - последнее совпадение на странице;

– :first-child - первый дочерний элемент;

– :last-child - последний дочерний элемент;

– :nth-child(n) - n-й элемент;

– :nth-child(even|odd) - четные или нечетные дочерние элементы;

– :even - четные элементы;

– :odd - нечетные элементы.

9. Выбор элементов на основе характеристик, не предусмотренных спецификацией CSS:

– :input - выбирает элементы формы;

– :selected - выбранные элементы option;

– :visible - выбор невидимых элементов.

Обработчик готовности документа

Первый способ - запуск обработчика, когда загрузилась страница window.onload=function(){$("table tr:nth-child(even) ").addClass("even");};.

Недостаток данного метода заключается в том, что ничего не будет изменено, пока все не загрузится, включая картинки, т.е. пользователь может увидеть, как все изменяется, т.е. исходный вариант и преобразованный.

Второй способ - дождаться только загрузки структуры документа, а именно: $(documen).ready(function(){

$("table tr:nth-child(even) ").addClass("even");

});.

Т.е. читаем буквально – запуск функции после загрузки‖. У этой конструкции имеется сокращенная форма:

```
$(function(){
$("table tr:nth-child(even) ").addClass("even");
});.
```

Создание элементов DOM

Создание элемента

$(' Hi ')

Пустой элемент можно создать так:

$(' ')

Можно использовать кавычки, можно апострофы. Но наличие тегов обязательно. Задавать просто текст нельзя. Сама по себе эта команда ничего не выведет, надо эту веточку опять же привязать к родительскому элементу. Делается это так:

$(' Hi ').insertAfter('#a1')

Здесь a1 - это идентификатор объекта, после которого необходимо вставить элемент.

Работа с полученным набором значений

В терминах jQuery эти наборы называют обернутыми.

Определение размера. Для этого можно использовать свойство length, а можно метод size().

Пример. Замена содержимого с идентификатором a1 на количество элементов класса b2 $("#a1").html($(".b2").size())

Полученный набор значений рассматривается как массив, и с ним мы, соответственно, и работаем. Получить любой элемент в обернутом наборе можно по индексу. Например, получить второй элемент в наборе всех гиперссылок: $('a')[1].

Вместо индексов можно использовать метод get(), т.е. последнюю команду можно записать так: $('a').get(1).

При помощи метода get() можно получить обычный массив JavaScript, содержащий все обернутые элементы. Пример: var allanchors=$('a').get().

Существуют специальные команды библиотеки, которые позволяют объединять различные полученные наборы, убирать определенные значения по некоторым правилам

Манипулирование объектами на странице

Команды jQuery позволяют манипулировать свойствами, атрибутами, стилями и содержанием элементов.

Для обращения к свойствам и их значениям используются методы JavaScript, нет методов непосредственно библиотеки.

Пример. Изменяет свойство title у всех элементов класса b2 $('.b2').each(function(n){this.title="New "+n})

Здесь использована команда **each (функция).** Она выполняет обход всех элементов в наборе и вызывает для них функцию. В качестве параметра функции передается индекс элемента в наборе.

Обработка событий

Различные браузеры по-своему могут обрабатывать события, jQuery пытается сгладить эти неприятности. Поэтому обращаемся только к методам jQuery, а библиотека уже сама смотрит, что за браузер и применяет то или иное свойство.

Модель событий jQuery обладает следующими свойствами:

– поддерживает единый метод установки событий;

– позволяет устанавливать несколько обработчиков для события;

– использует стандартные названия типов событий;

– предоставляет единые методы отмены события и блокирования действий по умолчанию.

Подключение обработчиков. Рассмотрим пример функции, которая будет срабатывать при щелчке по любому рисунку:
$('img').blind('click',fuction(event){alert('Приветствую!');}); Для удаления обработчика используется команда unbind()

Скрытие и отображение объектов

Функции **hide**() и **show**(). С ними есть некоторые нюансы (например, show показывает изначальное состояние свойства display, поэтому при загрузке рекомендуют скрыть объект при помощи hide()).

Имеются различные эффекты – раскрывающиеся списки, слайдеры, увеличение рисунков (когда-нибудь надо разобрать подробнее)

Создания слайдера

Рассмотрим пример создания слайдера - блока с контентом, выползающего по нажатию на конец закладки.
В текст вставляем следующий фрагмент:

```
<div id="container">
<div id="panel"> Содержимое панели
</div>
<p class="button"><a href="#" class="buttontext">Справка</a></p>
</div>
```

При помощи стилей изначально скрываем панель. Остальные параметры просто описательные - размер, цвет и т.д.

```
<!--Из-за наличия auto заодно и центрует в Mozila, т.к. auto Указывает, что размер отступов будет автоматически рассчитан браузером.-->
<!--buttontext можно вообще не задавать-->
<style>
.rrr             {color:red}
#container {
margin: 0 auto; width: 152px;
}
#panel {
```

```
background: #1ca8f6; height: 230px; display: none;
}
.button {
width: 152px; height: 40px;
border-top: #333 dotted 1px; text-align: center;
}
.buttontext {
font-weight: bold; font-size: 1.2em;
text-shadow: 1px 1px 1px #666;
}
</style>
```

Описываем функцию

```
<script type="text/javascript">
$(document).ready(function(){
$(".buttontext").click(function(){
$("#panel").slideToggle("normal"); return false;
});
});
</script>
```

Комментарии

– Команда **.slideToggle** позволяет чередовать два простых действия: показать и скрыть. normal - это скорость, означает, что панель выедет за 400 миллисекунд, есть еще fast, slow.

– Команда **return false** означает, что мы не даем перейти по ссылке, в качестве которой используется кнопка-закладка. То есть она просто работает в качестве удобной ручки, за которую дергают, но не как непосредственно ссылка.

РАЗДЕЛ 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

Во втором разделе представлен комплекс практических работ.

Практическая работа № 1
Основы языка HTML.

Цель работы: изучить основы языка HTML. Создать домашнюю страницу.

Задание : Создание домашней страницы содержащей информацию о студенте.

Обязательное содержание:

- Информация о том, чей сайт и фото студента.
- Электронный почтовый адрес.
- Ссылка на рабочую страницу и сайт института.
- Мой ВУЗ
- Моя группа.
- Моя будущая профессия.
- Мои увлечения или хобби.
- Любая другая информация.

В оформлении страницы следует использовать максимальное количество вышеперечисленных тегов HTML.

Контрольные вопросы:

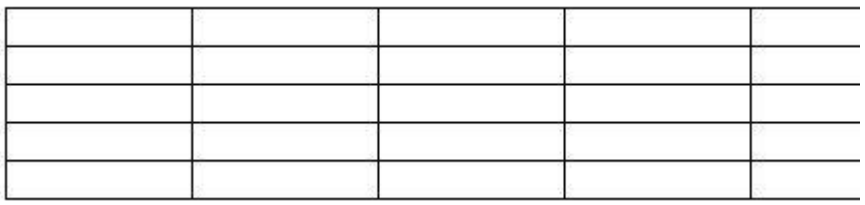
1. Почему в браузере не отображается текст, расположенный между тегами <!-- и -->?
2. Какое утверждение наиболее подходит к случаю, когда тег <p> располагается в контейнере <head>?
3. Дайте определение элементу main. Какова его цель?

Практическая работа № 2
Оформление HTML документа. Таблицы

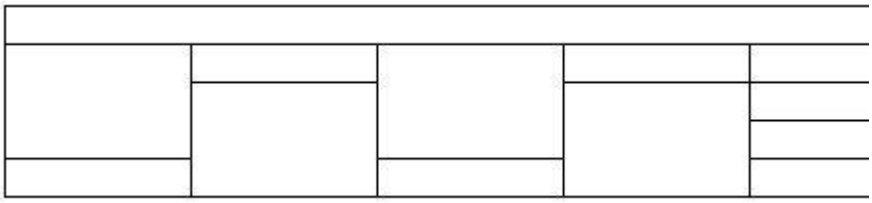
Цель работы: изучить основы организации таблиц в HTML.

Задания:

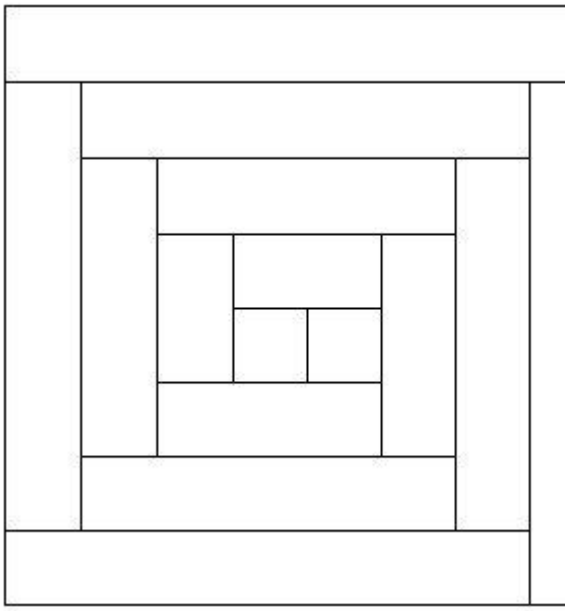
1. Постройте таблицу следующего вида:



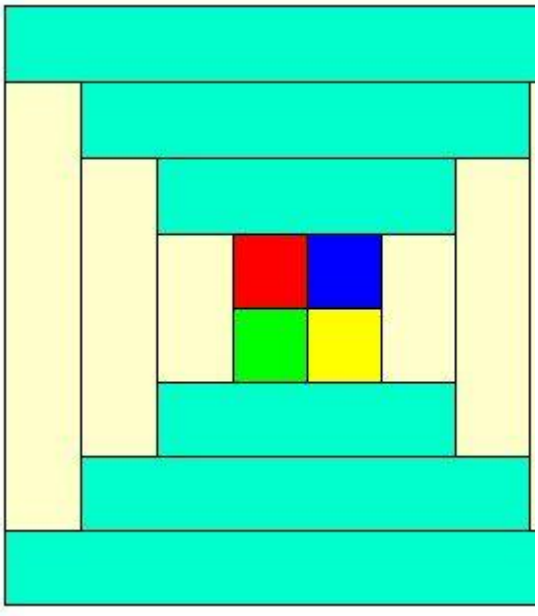
2. Постройте таблицу следующего вида:



3. Постройте таблицу следующего вида:



4. Постройте таблицу следующего вида:



5. Использование стандартного параметра BORDER приводит к неоднозначности отображения границы таблицы. Точнее каждый браузер интерпретирует вид границы по-своему. Для устранения подобного

разночтения создают таблицу с простой однопиксельной рамкой. Последовательность действий такова:

1. Создать таблицу и "подложку", состоящую из одной строки и одного столбца. Залить ее требуемым цветом.
2. Далее в эту таблицу помещаем вложенную, заливая каждую *ячейку* нужным цветом.
3. Устанавливаем свойство CELLSPACING вложенной таблицы равным 1, тогда нижняя таблица будет просвечивать и образует рамку толщиной 1 пиксель.

Создайте таблицу 6x6 с простой однопиксельной рамкой.

Контрольные вопросы:

1. Какой функцией можно изменить размер ячейки?
2. Чем отличается divot span?
3. Как обозначаются комментарии в html?

Практическая работа№3
Размещение скриптов в HTMLдокументе

Цель работы: изучить основы размещения скриптов в HTML документе

Задание 1.

1. Создайте простой HTML-документ.
2. Добавьте два абзаца с произвольным текстом.
3. Организуйте между двумя абзацами вывод приветственного сообщения в диалоговом окне, задав необходимые команды внутри тэга <script>.
4. Добавьте команду вывода аналогичного приветственного сообщения в окно браузера после закрытия диалогового окна.
5. Сохраните документ с именем Ex1.html в рабочей папке.

Задание 2.

1. Создайте простой HTML-документ.
2. Добавьте два абзаца с произвольным текстом.
3. Организуйте между двумя абзацами вывод приветственного сообщения в диалоговом окне, задав необходимые команды JavaScript во внешем файле. Для этого:
   - создайте новый текстовый файл,
   - поместите в него код JavaScript,
   - сохраните файл с именем main.js следующим образом: укажите тип файла “Все файлы”, кодировку “UTF-8”.
4. Добавьте ссылку на внешний скриптовый файл из рабочего HTMLдокумента.
5. Сохраните документ с именем Ex2.html в рабочей папке.

Задание 3.

1. Создайте простой HTML-документ.

2. Сохраните документ с именем Ex3.html в рабочей папке**.**

3. Добавьте в документ код JavaScript так, чтобы в диалоговом окне появлялось поле с надписью "Введите сюда своё имя" и со значением по умолчанию в поле "Введите имя". Для этого используйте метод promt(...) объекта window. Для хранения введенного значения заведите новую переменную.

4. Организуйте вывод введенного значения имени в окно браузера в виде: "Ваше имя <…..>".

5. Дополните код, чтобы в новом диалоговом окне появилось надпись "Начать заново? " При положительном ответе должно появиться диалоговое окно: "Не надоело? ", при отказе – "Ну и правильно!". Используйте для написания методы alert(…) и confirm(…) объекта window.

Контрольные вопросы:

1. Как сделать ссылку на сайт?
2. Что делают теги <EM></EM>?
3. Что такое ENTITIES?

Практическая работа №4
Построение системы html-документов и их оформление при помощи CSS

Цель работы: научиться создавать систему html-документов, освоить оформление html-документов при помощи каскадных таблиц стилей.

Задание: Создайте систему html-документов, соединенных друг с другом ссылками.

Разработанные документы оформите при помощи стилей CSS.

Ход работы

Создайте пять html-документов. Изучите основы стилей CSS (http://htmlbook.ru/samcss) и оформите созданные html-документы при помощи CSS. Существует три способа подключения стилей CSS к html-документу:

1. описание в атрибуте style тегов:
<div style=”border: 1px solid gray;”>… ;
2. описание в теге style, расположенном в заголовкедокумента:
<head>
…
<style>
div {
}
</style>
…
</head>
border: 1px solid gray; float:
left;
3. подключение из внешнего css-файла при помощи тегalink:
<linkrel="stylesheet" type="text/css" href="style.css">

при этом style.css содержит описание правил css, например:

```
div{
border: 1px solid gray;
float: left;}
#menu div {
margin: 10px;
background: #CCCCCC url(./path/to/image.jpg) repeat-x top left;}
```

Используйте все три способа подключения CSS.

Для оформления документов используйте следующие свойства CSS:

- font-family, font-size, font-weight, color,text-decoration;
- margin,padding;
- background-color, background: #FFFFFF url(./path/to/image.jpg) repeat-x topleft;
- border: 1px solid gray; для таблиц border-collapse (separate,collapse);
- для div: display (block, inline), float, position (absolute, relative).

Справочник свойств CSS – http://htmlbook.ru/css.

В верхней части каждого документа должно быть расположено меню для перехода между документами. Переходы между документами сделайте при помощи относительных и абсолютных ссылок:

(http://htmlbook.ru/samhtml/ssylki/absolyutnye-i-otnositelnye-ssylki).

При помощи Firebug изучите элементы Ваших страниц (в правой части окна Firebug вкладки Стиль, Скомпилированный стиль, Макет, DOM). Найдите соответствие между атрибутами тегов во вкладке HTML основного окна Firebug и их свойствами во вкладке DOM (пояснения - http://javascript.ru/tutorial/dom/attributes).

Контрольные вопросы:

1. Язык CSS. Примеры описания каскадных таблиц стилей.
2. CSS-атрибуты блоков: границы, отступы
3. CSS-трансформации и переходы.
4. CSS-атрибуты текста.

Практическая работа №5

JavaScript. Динамическое изменение html- документа в браузере

Цель работы: изучить основы javascript и научиться динамически изменять html- документ.

Задание: Создать html-документ, сделать динамическое создание списка группы.

Ход работы

Создайте html-документ и вставьте в него нумерованный список с атрибутом id, равным list, затем добавьте две строчки.

```
<ponclick=”add_to_list()”> Добавить в список</p>
<ulid=”list”></ul>
```

onclick=”add_to_list()” означает, что для данного элемента <p> задаем javascript событие, которое будет срабатывать при клике на элемент, а в тегах <ul> будет находиться будущий список.

Добавьте в <head> тег <script> с атрибутом type равным "text/javascript". Здесь можно уже вставлять javascript код. Создайте массив, содержащий в себе список группы.

var arr = ["Ivanov", "Petrov", "Sidorov"];

Так же необходимо создать переменную для счета элементов в массиве.

varcounter = 0;

Теперь настало время для определения функции add_to_list(). Алгоритм работы этой функции следующий:

1. Проверить, есть ли в массиве запись, если нет, то выйти из функции. Это делается с помощью условия if (typeof(arr[counter]) != "undefined"). Функция type of() возвращает тип данных объекта, находящегося в скобках. В данном случае тип данных элемента массива.

2. Получить список, куда мы будем добавлять созданный элемент, и присвоить его переменной list. Для выполнения этого пункта необходимо дать понятие о DOM (documentobjectmodel) - объектная модель, используемая для XML/HTML- документов. Согласно DOM-модели документ является иерархией. Каждый HTML-тег образует отдельный элемент-узел, каждый фрагмент текста текстовый элемент и т.п. Проще говоря, DOM - это представление документа в виде дерева тегов. Это дерево образуется за счет вложенной структуры тегов плюс текстовые фрагменты страницы, каждый из которых образует отдельный узел.

Для манипуляций с DOM используется объект document и функции:

– getElementById('el_id') – возвращает элемент дерева с id равным el_id;

– getElementsByTagName('li') – возвращает массив элементов дерева, тэгом которых является тэг li;.

– getElementsByName(‘el_name’) – возвращается массив элементов дерева, у которых атрибут name равен el_name;

– createElement(‘element’) – функция для создания нового элемента, где element это тот элемент, который необходимо создать;

– appendChild(‘element’)– добавляет в DOM новый элемент.

– varlist = document.getElementById('list') – получает элемент с id равным list и присваивает объект в переменную.

3. Создать элемент li и присвоить его переменной li. Чтобы создать элемент необходимо воспользоваться функцией createElement('element'), где element – это тот элемент, который необходимо создать.

var li = document.createElement('LI')

4. Добавить в него текст. Для этого используется свойство innerHTML, которое позволяет вставить код внутрь элемента.

li.innerHTML = arr[counter]

5. Вставить элемент в конец списка при помощи метода appendChild. list.appendChild(li)

6. Увеличить счетчик. counter++.

Контрольные вопросы:

1. Что такое объект (класс)? Экземпляр объекта?
2. Что означают термины «свойства объекта», «методы объекта»?
3. Что означает термин «инкапсуляция»?
4. Что понимают под интерфейсом объекта?

Практическая работа №6
Работа с сессиями. Реальная авторизация и регистрация

Цель работы: научиться работать с сессиями.

Задание: Создание авторизации и регистрации.

Ход работы

Веб-сервер не поддерживает постоянного соединения с клиентом, и каждый запрос обрабатывается, как новый, безо всякой связи с предыдущими. То есть, нельзя ни отследить запросы от одного и того же посетителя, ни сохранить для него переменные между просмотрами отдельных страниц. Для решения этих двух задач и были изобретены сессии.

Собственно, сессии, если в двух словах - это механизм, позволяющий однозначно идентифицировать браузер и создающий для этого браузера файл на сервере, в котором хранятся переменные сеанса.

Теперь создайте файл registration.php, который будет содержать форму регистрации пользователя.

```
<form method="POST" action="registration.php">
<p>Имя пользователя: <input type="text" name="username"></p>
<p>Email: <input type="text" name="email"></p>
<p>Пароль: <input type="password" name="password"></p>
<p>Пароль (повторить): <input type="password" name="password2"></p>
<p><input type="submit"></p>
</form>
```

В самом начале файла необходимо вызвать функцию session_start(), она создает сессию или продолжает текущую на основе текущего идентификатора сессии, который передается через запросы, такие как GET, POST или cookie. В большинстве случаев используют сессии на cookie, поэтому перед функцией start_session() не должно быть функций, возвращающих сообщение в браузер. Затем делаем проверку, аутентифицирован ли пользователь.

```
if ($_SESSION['username']) {
echo 'Вы уже зарегистрированы'; return 0;
}
```

Далее проверяем, существует ли POST запрос, если да, то проверяем, совпадают ли пароли, и заносим в переменную сессии данные из POST запроса.

```
if ($_POST) {
if ($_POST['password'] == $_POST['password2']) {
$_SESSION['username'] = $_POST['username'];
echo 'Пользователь '.$_SESSION['username'].' зарегистрирован'; return 0;
} else {
echo 'Введенные пароли не совпадают'; return 0;
}
```

```
}
```

И в index.php заменяем все POST на SESSION.

Теперь необходимо создать файл logout.php. Этот файл будет содержать функцию, которая разрушит все данные, зарегистрированные в сессии – session_destroy().

```
<?php
session_start(); session_destroy(); header('Location: index.php');
?>
```

Перед тем как отправить форму, ее нужно проверить. Проверка формы будет осуществляться посредством javascript. Для этого у формы определим событие onsubmit="returncheckForm(this)". Функция checkForm будет вызываться перед отправкой данных. Проверять будем имя пользователя, заполнено он или нет, и email. Шаблон email`а будет [английские_буквы]@ [английские_буквы]. [английские_буквы]. Такую проверку можно сделать, используя регулярные выражения – это формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании. По сути это строка-образец, состоящая из символов и метасимволов и задающая правило поиска.

Вообще регулярное выражение для проверки электронной почты довольно громоздкое.

```
^[-a-z0-9!#$%&'*+/=?^_`{|}~]+(?:\.[-a-z0-9!#$%&'*+/=?^_`{|}~]+)*@(?:[a-
z0-9]([-a-z0-9]{0,61}[a-z0-
9])?\.)*(?:aero|arpa|asia|biz|cat|com|coop|edu|gov|info|int|jobs|mil|mobi
|museum|name|net|org|pro|tel|travel|[a-z][a-z])$
```

Но оно слишком тяжело для понимания, поэтому воспользуемся простой формой [a- zA-Z]*@[a-zA-Z]*\.[a-zA-Z].

В начале функции определим все переменные, которые нам понадобятся, и массивы с ошибками.

```
varel, // Сам элемент elName, // Имя элемента формы value, //Значение
type; // Атрибут type для input-ов
reg = /[a-zA-Z]*@[a-zA-Z]*\.[a-zA-Z]/; varerrorList = [];
varerrorText = {: "Не заполнено поле'Имя'",
1  : "Не заполнено поле'E-mail'",}
```

Далее проходим по всем элементам формы и проверяем все теги input. Если поля для имени пусты или электронная почта не соответствует шаблону, записываем номера ошибок в массив.

```
for (vari = 0; i<form.elements.length; i++) { el = form.elements[i];
elName = el.nodeName.toLowerCase(); value = el.value;
if (elName == "input") {
type = el.type.toLowerCase(); switch (type) {
case "text" :
if (el.name == "name" && value == "")
errorList.push(1);
errorList.push(2);
if (el.name == "email" && !value.match(reg))
break; default : break;
}}}
```

Затем проверяем массив с ошибками. Если он пуст, то возвращаем true, иначе формируем текст для ошибки, выводм это на экран и возвращаем false.

```
if (!errorList.length) return true;
var errorMsg = "При заполнении формы допущены следующие ошибки:\n\n"; for (i = 0; i < errorList.length; i++) {
errorMsg += errorText[errorList[i]] + "\n";}
alert(errorMsg); returnfalse.
```

Контрольные вопросы:

1. Какую функцию следует использовать для проверки на ошибки?
2. Какая функция возвращает идентификатор текущего сеанса?
3. Какие открывающие и закрывающие дескрипторы лучше использовать?

Практическая работа №7

Гостевая книга на файлах

Цель работы: создание гостевой книги на файлах.

Задание: Создать гостевую книгу.

Ход работы

Первоначально нужно определить функционал гостевой книги, т.е. как она будет работать. Сообщения могут оставлять только зарегистрированные пользователи, а пользователи-гости будут видеть только сами комментарии и надпись, что нужно зарегистрироваться. Гостевая книга будет находиться в файле index.php. Все данные будут записываться в файл guestbook.txt.

Для создание гостевой книги необходима форма, которая будет добавлять сообщения, поэтому разместим ее в файле после поля для логина.

```
<form method="POST">
<p> Тема поста: <input type="text" name="theme"></p>
<p>Текст поста:<textarea rows="10" cols="45" name="text"></textarea></p>
<p><input type="submit"></p>
</form>
```

Затем запрос этой формы необходимо обработать. Поместим в начало файла после запуска сессии проверку, если существует POST запрос с параметрами theme, text и пользователь находится в сессии, то открываем или создаем файл guestbook.txt, записываем туда данные, закрываем файл и создаем переменную с сообщением "Комментарий был успешно добавлен".

```
if ($_POST['theme'] && $_POST['text'] && $_SESSION['username']) {
$fp = fopen('guestbook.txt', 'a+'); if (!$fp)
{
$fp = fopen('guestbook.txt', 'w+');
}
$mytext
=
'username='.$_SESSION['username'].'&text='.$_POST['text'].'&theme='.$_POST[' theme']."&"."\r\n";
$test = fwrite($fp, stripslashes($mytext));
```

```
fclose($fp);
        $msg = "Комментарий был успешно добавлен";
        }
```

Перед формой разместим условие, если `$msg` существует, вывести его на экран, и если пользователь не в сессии, то вместо формы выводим «Чтобы оставлять комментарии вы должны быть зарегистрированы».

```
        <?php
        if ($msg) {
        ?><p><?phpecho $msg; ?></p><?php
        }
        if ($_SESSION['username']) {
        ?>
        <form method="POST">
        <p> Тема поста: <input type="text" name="theme"></p>
        <p>Текст поста:<textarea rows="10"
           cols="45" name="text"></textarea></p>
        <p><input type="submit"></p>
        </form>
        <?php
        } else {
        ?><p>Чтобы оставлять комментарии вы должны быть
зарегистрированы</p>
        <?php
        }
        ?>
```

Далее следуют сами комментарии. Выводить их будем в таблице вида

имя пользователя	заголовок
Текст сообщения	

Для этого открываем файл guestbook.txt с параметром “r”, затем при помощи функций feof() и fgets() считываем построчно файл. foef() проверяет, достигнут ли конец файла, fgets() – возвращает строку из файла. Затем при помощи регулярных выражений вытаскиваем имя пользователя, заголовок, текст сообщения и заносим данные в таблицу.

```
        <tableclass="table">
        <?php
        $fp = fopen('guestbook.txt', 'r'); while
(!feof ($fp)) {
        ?>
        <tr class="main">
        <?php
        $buffer = fgets($fp); preg_match('/username=([^&]*)&/',
$buffer, $user); preg_match('/text=([^&]*)&/', $buffer, $text);
preg_match('/theme=([^&]*)&/', $buffer, $theme);
        ?>
        <td>
        <?php echo $user[1];?>
        </td>
        </tr>


        </td>
        <?php echo $theme[1];?>
        <tr class="text">
        <td colspan=2>
        <?php echo $text[1]; ?>
```

```
</td>
</tr>
<?php
}
fclose ($fp);
?>
</table>
```

В принципе гостевая книга готова, но чтобы было удобнее работать, ее нужно оформить при помощи css. Объединим поля для регистрации и входа в один div с id=reg.

```
<divclass="reg">
<ahref="registration.php">Регистрация</0a>
<ahref="login.php">Войти</a>
<p>Привет,
<?php?>
<?php
if ($_SESSION['username']) {
echo$_SESSION['username'];
<a href="logout.php">Выйти</a>
} else {
echo 'Гость';}
?>
</p>
</div>
```

Создадим и подключим файл guestbook.css.

Пример файла guestbook.css:

```
table {
width: 300px;}
.main{font-size: 120%;
font-family: Verdana, Arial, Helvetica, sans-serif; color: #336;
border: 10px solid #666;}
.text{color: red;
border: 1px solid #666;
background: #eee; padding: 5px;}
.reg{position: absolute; right: 10px;
top: 10px; width:
225px; height: 180px;
background: #f0f0f0;}
```

Контрольные вопросы:

1. Какие функции можно использовать для изменения цвета?
2. Каким образом можно подключить файл?

Практическая работа №8
Чтение и запись в файл. Регистрация с записью в файл. Авторизация из файла

Цель работы: научиться работать файлами.

Задание: Создание авторизации и регистрации на файлах.

Ход работы

Для работы с файлами в php существует несколько функций.

fopen– функция для открытия файла.

$fp = fopen('filename, 'param');

filename и param это обязательные параметры. Первый отвечает за имя файла, который необходимо открыть, а второй определяет режим файла:

1. r – открытие файла только для чтения.
2. r+ - открытие файла одновременно на чтение и запись.
3. w – создание нового пустого файла. Если на момент вызова уже существует такой файл, то он уничтожается.
4. w+ - аналогичен r+, только если на момент вызова файл такой существует, его содержимое удаляется.
5. a – открывает существующий файл в режиме записи, при этом указатель сдвигается на последний байт файла (на конец файла).
6. a+ - открывает файл в режиме чтения и записи при этом указатель сдвигается на последний байт файла (на конец файла). Содержимое файла не удаляется.

Записывать данные в файл при помощи PHP можно при помощи функции fwrite(). Это функция принимает 2 обязательных параметра и 1 необязательный. В качестве обязательных параметров выступает дескриптор файла и режим файла:

$test = fwrite($fp, $mytext);

По завершению работы с файлом его нужно закрыть, используя функцию

fclose($fp).

Теперь в файле registration.php после строки if '($_POST) {' поставим проверку файла, если он существует, то открываем его и перемещаем указатель в конец строки, если нет, то создаем его.

```
$fp = fopen('users.txt', 'a+'); if (!$fp) {
$fp = fopen('users.txt', 'w+');
}
```

В условие проверки паролей сформируем строку с данными, которые будут разделены знаком &, запишем ее в файл и закроем его.

```
$mytext = 'username='.$_POST['username'].'&password='.$_POST['password'].'&email
='.$_POST['email']."\r\n";
$test = fwrite($fp, $mytext); fclose($fp);
```

Следующим шагом будет преобразование файла login.php. Теперь мы будем получать данные о пользователях не в самом сценарии, а из отдельного файла. Для этого в самом начале файла снова начнем сессию, подключим файл debug.php для отладки и делаем проверку: если существует пост запрос, то открываем файл user.txt, построчно считываем его содержимое и сравниваем с данными POST запроса. В случае успеха добавляем пользователя в сессию и переходим на index.php.

```
<?php session_start();
require_once('debug.php'); if ($_POST) {
$fp = fopen('users.txt', 'r'); while (!feof ($fp)) {
$buffer = fgets($fp); preg_match('/username=([^&]*)&/', $buffer, $user);
preg_match('/password=([^&]*)&/', $buffer, $pass);
if ($_POST['username'] == $user[1] && $_POST['password']
==
```

```
$pass[1]) {
$_SESSION['username'] = $_POST['username']; header('Location: index.php');
}
}
fclose ($fp);
} else {
?>
<html>
<head>
<title>Login page</title>
</head>
<body>
<h3>Войти</h3>
<form method="POST" action="">
<p>Имя пользователя: <input type="text" name="username"></p>
<p>Пароль: <input type="password" name="password"></p>
<p><input type="submit"></p>
</form>
<?php
}
?>
</body>
</html>
```

Контрольные вопросы:

1. С помощью какой функции можно открыть файл?
2. За что отвечает функция fwrite()?
3. Как можно сформировать строку?

Практическая работа №9

Операторы управления, функции. Объекты ядра JavaScript

Цель работы: изучить операторы управления и функции.

Задание 1.

1. Рассмотрите пример скрипта:

```
<html>
<head>
<title>if</title>
</head>
<body>
<script language="JavaScript" type="text/JavaScript"> var x, y;
x=parseInt(prompt("Введите значение x",")); // метод parseInt()
переводит строку в целое y=parseInt(prompt("Введите значение y",")); // число if(x<y) {
alert("Максимальное число - y")
}
else {
alert("Максимальное число - x")
}
</script>
</body>
```

</html>

2. Допишите скрипт так, чтобы при введении пользователем одинаковых чисел открывалось сообщение "Введенные числа равны!".

3. Напишите скрипт, в котором пользователя просят ввести правильный пароль. При вводе правильного пароля в окне браузера появляется сообщение о том, что пароль верен. При вводе неправильного пароля – выпадает сообщение о неправильно введенном пароле. Для выполнения задания введите переменную password, в которую сохраните верное значение пароля.

4. Сохраните документ с именем Ex4.html в рабочей папке.

Задание 2.

1. Рассмотрите пример скрипта:

```
<html>
<head>
<title>for</title>
</head>
<body>
<h1>Пример простой</h1>
<scriptlanguage="JavaScript"
type="text/JavaScript">functionline() {
document.writeln("<hralign='center' width='100'>");
}
for (var i=1; i<10; i++)
line(); </script>
</body>
</html>
```

2. Создайте вариант прорисованных линий со следующими условиям:

- десять линий должны располагаться друг под другом;
- первая должна быть длинной 10 пикселей;
- каждая последующая на 10 пикселей больше.

3. Сохраните документ с именем Ex5.html в рабочей папке.

Задание 3.

Создайте простой HTML-документ.

1. Сохраните документ с именем Ex6.html в рабочей папке**.**

2. Добавьте в документ код JavaScript так, чтобы в окне браузера была выведена таблица степеней двойки следующего вида:

Степень	Результат
2^0	1
2^1	2
2^2	4

2^3	8
2^4	16
2^5	32

Для этого в сценарии используйте метод write(…) объекта document для формирования содержимого страницы. На каждой итерации цикла for сформируйте очередную строку таблицы, в первую ячейку которой заносится соответствующая степень двойки, а во вторую – результат ее возведения в указанную степень. Для выполнения этого действия используется встроенный объект Math и его метод pow(…), возводящий первый параметр в степень, заданную вторым параметром. Обратите внимание, что метод write(…) может вызываться с любым количеством фактических параметров. Результатом его работы в любом случае является вывод в документ строки, полученной конкатенацией всех параметров, переданных в метод.

Задание 4.

1.Рассмотрите пример скрипта:

```
<html>
<head>
<title>array</title>
</head>
<body>
<script language="JavaScript">
   year=new Array("декабрь","январь","февраль","март","апрель","май",
"июнь","июль","август","сентябрь","октябрь","ноябрь
"); summer=new Array();     //летние месяцы
summer=year.slice(6,9); document.write(summer+"<br>");
</script>
</body>
</html>
```

2. Создайте массив, содержащий названия школьных предметов. Выделите из него два массива. Пусть к первому относятся предметы из раздела точных наук, а ко второму - из раздела гуманитарных наук. Для создания и вывода в окно браузера новых массивов используйте метод slice(…) и wtite(…) объекта document. Оформите исполняющий скрипт в виде отдельной функции, описанной в разделе <head> и вызванной в разделе <body>.

3. Сохраните документ с именем Ex7.html в рабочей папке**.**

Задание 5.

Создайте простой HTML-документ.

1. Сохраните документ с именем Ex8.html в рабочей папке**.**

2. Добавьте скрипт, на основе которого будут выполняться следующие условия:

– если на страницу зашел пользователь через браузер MicrosoftInternetExplorer, перенаправьте его автоматически на страницуEx1.html;

– если на страницу зашел пользователь через любой другой браузер, перенаправьте его на страницу Ex3.html.

Для выполнения задания используйте свойство appName объекта navigator.

Контрольные вопросы:

1. Что такое Hoisting?
2. Что означает функция alert?
3. Что означает parseInt?

Практическая работа №10
Объекты клиентских приложений. Обработка событий

Цель работы: научиться работать с объектами клиентских приложений.

Задание 1.

Рассмотрите скрипт:

```
<html>
<head>
<title>document</title>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
document.write("Спасибо, что пришли к нам на курсы!");
</script>
</body>
</html>
```

1. Допишите скрипт так, чтобы

– цвет фона документа был #E7E6D8,

– цвет шрифта – красный,

– внизу выводилась дата последней модификации документа, используйте для этого слияние методов wtite(…) и lastModified(…) объекта document.

2. Сохраните документ с именем Ex9.html в рабочей папке.

Задание 2.

1. Рассмотрите пример скрипта открытия нового окна на странице:

```
<html>
<head>
<title>window</title>
</head>
<body>
<h1>Создание нового окна</h1>
<hr>
```

```
<script language="JavaScript" type="text/JavaScript">
window.open("http://www.google.com","","toolbar=no,scrollbars=yes,widt
h=250, height=250, resizable=yes, top=100, left=500")
</script>
</body>
</html>
```

2. Измените скрипт так, чтобы выполнялись следующие условия:

– открытие нового окна происходило при нажатии на ссылку с текстом: «Щелкните на ссылке для получения справочной информации»,

– размеры окна - 500x500,

– есть возможность изменения размеров окна.

Для выполнения задания используйте написание функции.

3. Сохраните документ с именем Ex10.html в рабочей папке.

Задание 3.

Создайте страницу с переадресацией на другой адрес (redirect).

1. Измените скрипт так, чтобы переадресация на другой адрес была с задержкой в 5 секунд.

2. Сохраните документ с именем Ex11.html в рабочей папке.

Задание 4.

Создайте HTML-документ, в котором будет две ссылки:

– первая ссылка должна ссылаться на PDF файл; при нажатии на нее выпадает сообщение с предупреждением о том, что для загрузки документа требуется программа Acrobat, и вопросом продолжить загрузку или нет; используйте для написания метод confirm(…) для подтверждения загрузки;

– вторая ссылка должна содержать такой код, чтобы при наведении на нее мыши менялся цвет фона документа на красный.

2. Сохраните документ с именем Ex12.html в рабой папке.

Задание 5.

Создайте HTML-документ, содержащий любую картинку.

1. Добавьте скрипт с условиями:

– при наведении курсора мыши на картинку она увеличивается;

– при отведении курсора мыши – уменьшается до исходного размера.

Постройте скрипт через использование функций и событий MouseOver и MouseOut.

2. Сохраните документ с именем Ex13.html в рабочей папке.

Задание 6.

Создайте HTML-страницу содержащую следующую форму заполнения данных:

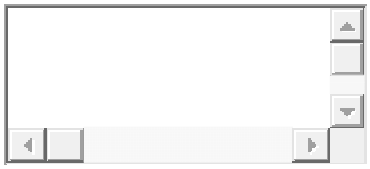
Ваше имя*

Пароль *

Подтверждение пароля*

Электронный адрес: *

Тема сообщения:
Сообщение:



Отправить Очистить

* – необходимые для заполнения поля

1. Добавьте скрипт, проверяющий следующие данные:

– заполнено ли поле имени:

– введен ли пароль и содержит ли он больше 4-х символов. Используйте для этого свойство length данного поля:

– совпадают ли значения, введенные в оба поля для паролей;

– заполнено ли поле электронного адреса и содержит ли оно символ @;

– заполнено ли поле сообщения и содержит ли оно больше 10 символов.

2. При несоблюдении условий курсор должен установиться в то поле, где пользователем введено неверное значение.

3. Сохраните документ с именем Ex15.html в рабочей папке.

Контрольные вопросы:

1. Какой скрипт нужно использовать, чтобы открыть новое окно?
2. Какой скрипт отвечает за переадресацию?

Практическая работа №11
Функция и обработка события

Цель работы: научиться работать с функцией при разработке приложений.

Основным элементом языка JavaScript является функция. Описание функции имеет вид

function F (V) {S},

где F - идентификатор функции, задающий имя, по которому можно обращаться к функции; V - список параметров функции, разделяемых

запятыми; S - тело функции, в нем задаются действия, которые нужно выполнить, чтобы получить результат. Необязательный оператор return определяет возвращаемое функцией значение. Обычно все определения и функции задаются в разделе <head> документа. Это обеспечивает интерпретацию и сохранение в памяти всех функций при загрузке документа в браузер.

Пример 1. Нахождение площади треугольника.

В предыдущих примерах пользователю не предоставлялась возможность вводить значения и в зависимости от них получать результат. Интерактивные документы можно создавать, используя формы. Предположим, что мы хотим создать форму, в которой поля Основание и Высота служат для ввода соответствующих значений. Кроме того, в форме создадим кнопку Вычислить. При щелчке мышью по этой кнопке мы хотим получить значение площади треугольника. Действие пользователя (например, щелчок кнопкой мыши) вызывает событие. События в основном связаны с действиями, производимыми пользователем с элементами форм HTML. Обычно перехват и обработка события задается в параметрах элементов форм. Имя параметра обработки события начинается с приставки on, за которой следует имя самого события. Например, параметр обработки события click будет выглядеть как onclick.

Листинг 1. Реакция на событие Click

```
<HTML>
<HEAD>
<title>Обработка значений из формы</title>
<script language="JavaScript">
<!--//
function care (a, h)
{
var s=(a*h)/2;
document.write ("Площадь прямоугольного треугольника равна ",s);
return s
}
//-->
</script>
</HEAD>
<BODY>
<P>Пример сценария со значениями из формы</P>
<FORM name="form1">
Основание: <input type="text" size=5 name="st1"><hr>
Высота: <input type="text" size=5 name="st2"><hr>
<input type="button" value=Вычислить
```

```
onClick="care(document.form1.st1.value, document.form1.st2.value)"> /*По клику мыши на кнопке в функцию care передаются два параметра - содержимое полей ввода*/
</FORM>
</BODY>
</HTML>
```

При интерпретации HTML-страницы браузером создаются объекты JavaScript. Взаимосвязь объектов между собой представляет иерархическую структуру. На самом верхнем уровне иерархии находится объект windows, представляющий окно браузера. Объект windows является предком или родителем всех остальных объектов. Каждая страница кроме объекта windows имеет объект document. Свойства объекта document определяются содержимым самого документа: цвет фона, цвет шрифта и т. д. Для получения значения основания треугольника, введенного в первом поле формы, должна быть выполнена конструкция

document.forml.stl.value,

т.е., говоря русским языком (при этом читаем с конца), используем данные value из поля ввода с именем st1 находящегося на форме form1 объекта document.

Пример 2. Вычисление площади квадрата

Напишем сценарий, определяющий площадь квадрата по заданной стороне. Площадь должна вычисляться в тот момент, когда изменилось значение его стороны. Пусть форма содержит два текстовых поля: одно для длины стороны квадрата, другое для вычисленной площади. Кнопка Обновить очищает поля формы. Площадь квадрата вычисляется при возникновении события change, которое происходит в тот момент, когда значение элемента формы с именем num1 изменилось и элемент потерял фокус. HTML-код представлен в примере 2.

Листинг 2. Реакция на событие Change

```
<HTML>
<HEAD>
<title>Обработка события Change - изменение значения элемента</title>
<script>
function srec(obj)
{obj.res.value=obj.num1.value* obj.num1.value}
</script>
</HEAD>
<BODY>
```

```
<P>Вычисление площади квадрата</P>
<FORM name="form1">
Сторона: <input type="text" size=7 name="num1"
onChange="srec(form1)">
<hr>
Площадь: <input type="text" size=7 name="res"><hr>
<input type="reset" value=Обновить>
</FORM>
</BODY>
</HTML>
```

Событие Focus возникает в момент, когда пользователь переходит к элементу формы с помощью клавиши <Tab> или щелчка мыши. Событие "потеря фокуса" (Blur) происходит в тот момент, когда элемент формы теряет фокус. Событие select вызывается выбором части или всего текста в текстовом поле. Например, щелкнув дважды мышью по полю, мы выделим поле, наступит событие select, обработка которого приведет к вычислению требуемого значения. В табл.3 представлены события и элементы документов HTML, в которых эти события могут происходить. В языке JavaScript определены некоторые стандартные объекты и функции, пользоваться которыми можно без предварительного описания. Одним из стандартных объектов является объект Math. В свойствах упомянутого объекта хранятся основные математические константы, а его методы можно использовать для вызова основных математических функций. В табл.4 приведены некоторые методы объекта Math. Выражение y=log x запишется y=Math.log(x).

Задания

1. Проверить примеры из лабораторной работы.
2. На плоскости заданы координаты трех точек. Напишите сценарий, который вычисляет площадь треугольника (использовать событие Focus).
3. Напишите сценарий, который для точки, заданной координатами на плоскости, определяет расстояние до начала координат (использовать событие Select).
4. Напишите сценарий, который обменивает местами значения двух введенных переменных (использовать событие Blur).

Практическая работа №12
Организация ветвлений в программах

Цель работы: научиться работать со структурой ветвление.

При составлении программы часто необходимо выполнение различных действий в зависимости от результатов проверки некоторых условий. Для организации ветвлений можно воспользоваться условным оператором,

который имеет вид: if B {S1} else {S2},
где B – выражение логического типа; S1 и S2 – операторы. Выполнение условного оператора осуществляется следующим образом. Вычисляется значение выражения B. Если оно истинно, то выполняются операторы S1, если ложно – операторы S2. Если последовательность операторов S1 или S2 состоит лишь из одного оператора, то фигурные скобки можно опустить. Возможна сокращенная форма условного оператора:

if B {S},

где B – выражение логического типа; S – последовательность операторов. Выполнение краткого условного оператора осуществляется так: вычисляется значение выражения B, если оно истинно, то выполняются операторы S.

Пример 1. Нахождение максимального значения

Для трех заданных значений a, b, c необходимо написать сценарий, определяющий максимальное значение. Поступим следующим образом. Сначала максимальным значением m будем считать значение a, далее значение b сравним с максимальным. Если окажется, что b больше m, то максимальным становится b. И, наконец, значение c сравнивается с максимальным значением из предыдущих значений a и b. Если c больше m, то максимальным становится c. Оператор присваивания

obj.res.value=m

обеспечивает запись вычисленного максимального значения в соответствующее поле формы. Функция Number (s) преобразует объект s, заданный в качестве параметра, в число. Полностью сценарий может быть записан так, как представлено в листинге 1.

Листинг 1. Вычисление максимального значения из трех заданных

```
<HTML>
 <HEAD>
 <TITLE>Вычисление максимального значения</TITLE>
 <script language="JavaScript">
 <!-- //
 function maxval (obj )
 {
 var a = Number(obj.num1.value);
 var b = Number(obj.num2.value);
 var c = Number(obj.num3.value);
 var m=a
 if (b > m) m=b
 if (c > m) m=c
 obj.res.value=m }
 //-->
 </script>
```

```
</HEAD>
<BODY>
<H4>Вычисление максимального значения</H4>
<FORM name="form1">
Число 1: <input type="text" size=8 name="num1"><hr>
Число 2: <input type="text" size=8 name="num2"><hr>
Число 3: <input type="text" size=8 name="num3"><hr>
Максимальное значение равно
<input type="button" value=Определить onClick="maxval(form1)">
<input type="text" size=8 name="res"><hr>
<input type="reset">
</FORM>
</BODY>
</HTML>
```

Решим рассмотренную задачу другим способом. Вспомним, что стандартный объект Math имеет метод max, который определяет наибольшее значение двух аргументов. Опишем функцию maxval1, которая определяет максимальное значение из трех заданных значений и использует объект Math.

```
function maxval1 (obj )
 {
 var a = Number(obj.num1.value);
 var b = Number(obj.num2.value);
 var c = Number(obj.num3.value);
 obj.res.value=Math.max(Math.max(a,b),c)
 }
```

Если бы требовалось определить максимальное из четырех заданных значений a, b, c, d, то можно было бы воспользоваться формулой

Math.max(Math.max(a,b), Math.max(c,d)).

Задания

1. Проверьте пример из лабораторной работы.
2. Вводится последовательность из пяти чисел. Напишите сценарий, в котором определяется число максимальных элементов.
3. Напишите программу, которая определяет, можно ли построить треугольник с заданными длинами сторон.
4. Точка на плоскости задается своими координатами. Определите, какой из четвертей прямоугольной системы координат принадлежит заданная точка.

Практическая работа №13
Методы в JavaScript

Цель работы: научиться применять методы в JavaScript.

Во время интерпретации HTML-документа браузером создаются объекты JavaScript. Свойства объектов определяются параметрами тегов языка HTML. Структура документа отражается в иерархической структуре объектов, соответствующих HTML-тегам. Родителем всех объектов является объект windows, расположенный на самом верхнем уровне иерархии, он представляет окно браузера и создается при запуске браузера. Для того чтобы открыть новое окно в сценарии JavaScript и отобразить в нем новый документ, применяется метод open, для закрытия окна можно воспользоваться методом close. Метод alert объекта windows отображает диалоговое окно с текстом, переданным методу в качестве параметра. Данный метод используется в случаях проверки правильности вводимых данных с помощью формы. Свойства объекта windows относятся ко всему окну, в котором отображается документ.

Подчиненными объектами (или объектами нижнего уровня) являются объекты document, history, location, frame. Свойства объекта history представляют адреса ранее загружаемых HTML-страниц. Свойства объекта location связаны с URL-адресом отображаемого документа, объекта frame – со специальным способом представления данных.

Свойства объекта document определяются содержимым самого документа: шрифт, цвет фона, формы, изображения и т. д. Объект document в зависимости от своего содержимого может иметь объекты, являющиеся для него подчиненными или дочерними. В частности, подчиненными для объекта document являются объекты form, image, link, area и др.

Для каждой страницы создается один объект document, некоторые его свойства соответствуют параметрам тега <BODY>: bgColor, fgcolor, linkcolor, alinkcolor, vlinkColor. Методы write и writeln записывают в документ текст, задаваемый параметром.

Если документ содержит изображения, то доступ к объекту, определяющему изображение, можно получить с помощью переменной, указанной в параметре name тега <Img>. Объект image имеет свойство images, которое содержит ссылки на все изображения, расположенные в документе. Ссылки перенумерованы, начиная с нуля. Доступ к первому изображению можно получить с помощью составной конструкции document.images[0], ко второму – document.images. Если на странице пять изображений, то доступ к последнему изображению можно получить, воспользовавшись ссылкой document.images.

Рассмотрим примеры, в которых используются различные свойства объектов.

Для встраивания изображений в HTML-документ служит тег <IMG>, имеющий обязательный параметр src, определяющий URL-адрес файла с изображением. Можно задавать размеры выводимого изображения. Значение

параметра width определяет ширину изображения, значение параметра height – высоту изображения. Значения параметров ширины и высоты могут не совпадать с истинными размерами изображений, тогда при загрузке изображения автоматически выполняется перемасштабирование.

Изображение можно поместить в рамку. Для этого используется параметр border. Значением параметра должно быть число, определяющее толщину рамки в пикселях. По умолчанию рамка вокруг изображения отсутствует, если только изображение не является ссылкой.

Параметр alt определяет альтернативный текст. При наведении курсора мыши на изображение появляется комментарий.

Пример 1. Перестановка изображений

Напишем сценарий, который реализует обмен рисунков в документе. Пусть в документе расположено три изображения, пронумерованных от 1 до 3. В текстовых полях указываются номера рисунков, которые необходимо поменять местами. Требуется, чтобы после нажатия кнопки Обменять изображения переместились на нужные места.

Сначала проверим, правильно ли заданы номера изображений, если это не так, то выдадим сообщение. Переменная z служит для запоминания адреса первого графического изображения. Доступ к изображению с номером r1 производится с помощью конструкции document.images[r1-1]. Для того чтобы на место изображения с номером r1 поместить изображение с номером r2, требуется выполнить оператор присваивания:

document.images[r1-1].src=document.images[r2-1].src

И, наконец, на место изображения с номером r2 помещается изображение, которое ранее было на месте с номером r1 и адрес которого запомнили в переменной z:

document.images[r2-1].src=z

Приведем полностью документ со сценарием (листинг 1).

Листинг 1. Перестановка изображений с помощью сценария

```
<HTML>
<HEAD>
<TITLE>Перестановка изображений</TITLE>
<script>
function chan(obj)
{ var r1=Number(obj.a1.value)
var r2=Number(obj.a2.value)
if ((r1<1)||(r1>3)||(r2<1)||(r2>3))
alert ("Неверно заданы номера рисунков!")
else
{ var z=document.images[r1-1].src
```

```
document.images[r1-1].src=document.images[r2-1].src;
document.images[r2-1].src=z
} }
</script>
</HEAD>
<BODY>
<CENTER>
<H4>Галерея рисунков</H4><br>
<IMG src="p1.gif" width="90" name="pic1">
<IMG src="p2.gif" width="90" name="pic2">
<IMG src="p3.gif" width="90" name="pic3"> <br>
<FORM name=form1>
Рисунки с номерами<br>
<input type="text" name="a1" size=1> и
<input type="text" name="a2" size=1><P>
<input type="button" value="Поменять местами" onClick="chan(form1)">
</FORM>
</CENTER>
</BODY>
</HTML>
```

Пример 2. Простое вертикальное меню

Напишем сценарий, реализующий вертикальное графическое меню. При наведении курсора мыши на пункт меню меняется цветовая палитра, соответствующая выделенному пункту меню.

Каждому пункту меню соответствует два изображения: первое изображение, когда пункт меню не выбран, второе - при выбранном пункте меню цветовая палитра рисунка изменена. Графические изображения, соответствующие ситуации, когда пункты меню не выбраны, хранятся в файлах с именами pch1.gif, pch2.gif. Соответствующие им графические изображения с измененной палитрой хранятся в файлах с именами wpch1.gif, wpch2.gif.

Функция img имеет два параметра. Первый параметр задает выбор пункта меню, второй параметр - n - определяет номер пункта меню. От этого параметра зависит, какое изображение в документе требуется изменить document.images[n-1].src (вставить на этом месте рисунок "wpch"+n+".gif" или pch"+n+".gif). Имя файла формируется динамически и представляет собой конкатенацию (слияние) строк, одна из составляющих которой значение второго параметра. Если имена файлов не подчинены общему правилу, то в функции потребуется дополнительный анализ, какой файл подгрузить. Это сделать нетрудно, зная место в документе, из которого произошел вызов функции. Документ со сценарием, реализующий вертикальное графическое меню, представлен в листинге 2.

Листинг 2. Простое вертикальное меню

```
<HTML>
<HEAD>
<TITLE>Простое вертикальное меню</TITLE>
<script language="JavaScript">
function img(n, action)
{if (action)
{document.images[n-1].src="wpch"+n+".gif"}
else
{document.images[n-1].src="pch"+n+".gif"}
}
</script>
</HEAD>
<BODY background="fon1.jpg">
<H2><FONT color="#0000ff">Содержание</FONT></H2>
<A href="tch1.htm" onmouseover="img(1,1)" onmouseout="img(1,0)">
<IMG src="pch1.gif" alt="форматирование текста" border="0" width="103" height="35"></A><br>
<A href="tch2.htm" onmouseover="img(2,1)" onmouseout="img(2,0)">
<IMG src="pch2.gif" alt="форматирование текста" border="0" width="103" height="35"></A><br>
<A href="tch3.htm" onmouseover="img(3,1)" onmouseout="img(3,0)">
<IMG src="pch3.gif" alt="форматирование текста" border="0" width="103" height="35"></A><br>
</BODY>
</HTML>
```

При попадании курсора мыши в область изображения возникает событие Mouseover, параметр обработки события onMouseOver получает значение img(p1,true).

Задания

1. Проверить примеры лабораторной работы.
2. Написать сценарий выбора из трех изображений одного, которое вставляется ниже этих трех.
3. Написать сценарий картинки с "эффектом приближения", т.е. увеличения размеров как реакция на попадание курсора мыши в поле рисунка (использовать свойства width и height).
4. Написать сценарий графического горизонтального меню с появляющейся стрелкой над пунктом, у которого находится курсор.

Практическая работа №14
Переключатели. Флажки. Списки

Цель работы: научиться применять элементы управления переключатель, флажки и списки.

Данные удобно представлять с помощью элемента управления "переключатель" (или "радиокнопка") в том случае, когда из нескольких вариантов может быть выбран лишь один.

Пример 1. Вычисление площади фигуры
Необходимо выбрать форму фигуры и определить ее площадь.
Пусть для выбора фигуры задана следующая форма:

```
<FORM name="form1">
Введите значение
<input type="text" name="data" size=10><hr>
Укажите форму:<br>
<input type="radio" name="fv" value=1>квадрат<br>
<input type="radio" name="fv" value=2>Круг<br>
<input type="radio" name="fv" value=3>треугольник<hr>
<input type="reset" value="Отменить"><hr>
Площадь: <input type="text" name="res" size=10>
</FORM>
```

В этой форме шесть элементов. Первый элемент служит для ввода строки текста. Следующие три элемента образуют группу и являются переключателями. Пятый элемент создает кнопку сброса, нажатие которой отменяет все сделанные изменения. Шестой элемент служит элементом для ввода строки.

Так как объект forms имеет свойство-массив elements, в котором содержатся ссылки на элементы формы в порядке их перечисления в теге <FORM>, то получить доступ к первому элементу формы можно либо с помощью значения параметра name этого элемента (document.form1.data), либо используя объектную модель JavaScript (document.forms[0].elements[0]). Второй элемент рассматриваемой формы можно получить, если воспользоваться конструкцией document.forms[0].elements[1]. Это элемент-переключатель, определенный в составе группы элементов. В рассматриваемом примере группа элементов состоит из трех переключателей. В одну группу входят элементы с одинаковым значением параметра name. Доступ к следующим элементам группы может быть осуществлен так: document.forms[0].elements[2], document.forms[0].elements [3]. Обязательный параметр value должен иметь уникальное значение для каждого элемента группы. Пользователь может выбрать только один вариант.

Напишем сценарий, в котором в зависимости от длины стороны или радиуса и формы выбранной фигуры вычисляется ее площадь. Для простоты будем считать, что фигура может иметь либо форму квадрата (задается его

сторона), либо форму круга (задается радиус), либо форму равностороннего треугольника (задается его сторона).

Площадь рассматриваемых фигур считается по формуле ka^2, где k - коэффициент, зависящий от формы выбранной фигуры; a - задаваемое пользователем значение. Вычисления будут проще, если коэффициент k указать в качестве значения параметра value соответствующего переключателя. Щелчок на элементе "переключатель" соответствует событию click, обработка которого заключается в вызове функции test. Функция имеет единственный параметр, значение параметра - value переключателя, которое служит для вычисления площади фигуры.

HTML-код приведен в листинге 1.

Листинг 1. Вычисление площади выбранной с помощью переключателя фигуры

```
<HTML>
<HEAD>
<TITLE>Данные из формы типа "переключатель". Событие Click </TITLE>
<script language="JavaScript">
<!--//
function test (k)
{ var a= form1.data.value
if (a !="" )
form1.res.value= k*Math.pow(a,2)
else alert ("Введите значение")
}
//-->
</script>
</HEAD>
<BODY>
<FORM name="form1">
Введите значение
<input type="text" name="data" size=10>
<hr>
Укажите форму <br>
<input type="radio" name="fv" value=1
onClick="test(form1.elements[1].value)">квадрат<br>
<input type="radio" name="fv" value=3.14
onClick="test(form1.elements[2].value)">круг<br>
<input type="radio" name="fv" value=0.42
onClick="test(form1.elements[3].value)">треугольник<hr>
<input type="reset" value="0тменить"><hr>
Площадь: <input type="text" name="res" size=10>
</FORM>
</BODY>
```

</HTML>

Пример 2. Выбор параметров обтекания изображения текстом

Напишем сценарий, который предоставляет возможность пользователю задавать значения параметров, определяющих, к какому полю окна (левому или правому) прижимается изображение и, соответственно, с какой стороны текст его обтекает.

Если значение параметра align равно Left, то изображение прижимается к левому краю окна просмотра браузера, а текст или другие элементы документа "обтекают" изображение с правой стороны. Текст, размещаемый рядом с изображением, может занимать несколько строк. По умолчанию значение параметра align равно Left. При нажатии на кнопку Обновить для изображения и текста будут установлены значения параметров, принимаемых по умолчанию.

Пример HTML-кода, который содержит сценарий, обеспечивающий выполнение действий, задаваемых пользователем, приведен в листинге 2.

Листинг 2. Обтекание текстом изображения

```
<HTML>
<HEAD>
<TITLE>Изображение и текст. Обтекание</TITLE>
<script>
<!--
function chpict(obj)
{
if ((obj.elements[0]).checked)
document.mypict.align="Left"
else
document.mypict.align="Right"
}
function rset(obj)
{document.mypict.align="Left"}
//-->
</script>
</HEAD>
<BODY>
<CENTER>
<H4>Изображение и текст.
Обтекание изображения текстом</H4>
</CENTER>
<FORM name="form1">
Выберите значение параметра выравнивания нажмите кнопку
<B>Просмотр</B>.<br>
```

```
<PRE>
<input type="radio" name="alg" checked value=ld>(left) изображение выравнивается по левому краю
<input type="radio" name="alg" value=rd>(right) изображение выравнивается по правому краю
</PRE>
<input type="button" value= "Просмотр" onclick="chpict(form1)">
<input type="reset" value="Отменить" onclick="rset(form1)"> </FORM>
<TABLE bgcolor="F8F8FF">
<TR><TD>Иван Иванович Шишкин является одним из основоположников
русского национального пейзажа.
<IMG src=pl.jpg name=mypict align=left border=3 width=310>
В полотне "Рожь" Шишкин создал образ большой эпической силы
и подлинно монументального звучания. Могучая, полная
богатырских сил природа, богатый, привольный край. (Т. Юрова)
</TD></TR>
</TABLE>
</BODY>
</HTML>
```

Если изображение рассматривается как элемент строки, то значения параметров выравнивания задают расположение изображения относительно строки текста. Верхняя граница изображения может быть выровнена либо по самому высокому текстовому элементу текущей строки, либо по самому высокому элементу в строке (например, другому изображению). Базовой считается нижняя часть линии текста, которая проводится без учета нижней части некоторых символов. Середину изображения можно выровнять либо по базовой линии, либо по середине текущей строки. Нижнюю часть изображения можно выровнять по базовой линии либо по нижней границе текущей строки.

Задания

1. Проверить примеры из лабораторной работы.
2. Напишите сценарий, который позволяет продемонстрировать изменения размеров и положения на странице горизонтальной линии.
3. Разработайте анкету, определяющую пол, возраст, семейное положение и т.п. человека.

Флажки

Элемент управления "флажок" используется в случае, когда из предложенных вариантов можно выбрать как один, так и несколько. Каждый вариант выбора задается флажком, который можно либо установить, либо сбросить. Флажок определяется в теге <input> значением checkbox параметра type. Обязательным параметром является параметр value, значение которого будет передано на обработку в случае выбора нажатием кнопки.

Пример 1. Выбор характеристик издания

Предположим, читателю предлагается заполнить анкету, в которой требуется указать название любимого издания и выбрать из предложенного списка характеристики, которые присущи рассматриваемому изданию.

Для задания характеристик издания можно воспользоваться флажком. Пользователь устанавливает флажки для тех свойств, которыми, по его мнению, обладает издание. Обработка анкеты будет состоять в том, что выбранные свойства будут отражены в поле ввода многострочного текста.

При щелчке мышью по флажку возникает событие click, обработка которого состоит в вызове функции set с одним параметром, принимающим значение параметра value флажка. Для формирования строки результата служит глобальная переменная s; к имеющемуся значению добавляется значение параметра функции и помещается в текстовое поле. Если нажать на кнопку Отмена, то очистятся все поля формы. Однако следует позаботиться о том, чтобы значение переменной s изменилось на начальное. Значение параметра реакции на событие click при щелчке по кнопке Отмена задается оператором присваивания, обеспечивающим начальные условия.

HTML-код представлен в листинге 1.

Листинг 1. Анкета читателя

```
<HTML>
<HEAD>
<TITLE>Анкета читателя</TITLE>
<script>
<!--
var s="Вас привлекает: \r\n"
function set(vch)
{ s=s+vch + "\r\n"; document.form1.area.value=s }
//-->
</script>
</HEAD>
<BODY bgcolor="F8F8FF">
<CENTER>
<H3 align="center">Анкета читателя</H3>
<FORM name="form0">
<H4>Введите название любимого журнала или газеты</H4>
<input type="text" name="n1" size=45><br>
</FORM>
<FORM name="form1">
<H4>Что Вас привлекает в издании?</H4>
<TABLE border=3 align=center> <TR>
<TD><img src="purpsm.gif" align="center"></TD>
```

```
 <TD><input type="checkbox" name="m1" value="Стиль подачи материала"
onClick="set(form1.elements[0].value)">
 Стиль подачи материала<br>
 <input type="checkbox" name="m2" value="Достоверность информации"
onClick="set(form1.elements[1].value)">
 Достоверность информации<br>
 <input type="checkbox" name="m3" value="Дизайн и оформление"
onClick="set(form1.elements[2].value)">
 Дизайн и оформление<br>
 </TD></TR></TABLE>
 <textarea name="area" cols=35 rows=7> </textarea><br>
 <input type="reset" value="Отмена"
 onclick= "s='Вас привлекает: \r\n'">
 </FORM>
 </BODY>
 </HTML>
```

В рассмотренных примерах значения параметра name флажков были различны, поскольку каждый флажок существовал независимо от других. Флажки можно объединить в группу. Для этого следует всем флажкам присвоить одно и то же значение параметра name.

Пример 2. Использование флажков в анкете переводчика

В анкете требуется указать те языки, которыми владеет переводчик. Предположим, что за знание каждого языка назначается определенная сумма. Размер вознаграждения определяется после заполнения анкеты в зависимости от тех языков, которыми пользователь владеет. По результатам заполненной переводчиком анкеты напишите сценарий определения размера вознаграждения.

Для задания сведений о том, владеет ли пользователь определенным языком, удобно применять флажок. При щелчке мышью по кнопке Вознаграждение выполняется функция grant(). Требуется проанализировать состояние флажков. Свойство checked возвращает логическое значение, представляющее текущее значение отдельного флажка (true или false). Воспользуемся тем, что каждый объект form имеет свойство-массив elements, получим доступ к каждому флажку формы. Состояние первого флажка можно определить с помощью следующей конструкции:

(document.forml.elements[0]).checked,

второго - (document. form1.elements[1]).checked

и т.д. В переменной к накапливается сумма. Шаг увеличения этой переменной задается в качестве значения параметра value. После анализа всех флажков полученная сумма выводится в документ.

HTML-код представлен в листинге 2.

Листинг 2. Данные, представленные флажком. Анкета переводчика

```
<HTML>
<HEAD>
<TITLE>Данные, представленные флажком. Анкета переводчика</TITLE>
<script language="JavaScript">
<!-- //
function grant()
{ var d= document
var k=0;
if ((d.form1.elements[0]).checked)
k=k+Number(d.form1.elements[0].value)
if ((d.form1.elements[1]).checked)
k=k+Number(d.form1.elements[1].value)
if ((d.form1.elements[2]).checked)
k=k+Number(d.form1.elements[2].value)
form1.ww.value="Вам полагается вознаграждение "+k+" у.е."
}
//-->
</script>
</HEAD>
<BODY>
<H3>Анкета для переводчиков</H3>
Укажите те языки, которыми Вы владеете в совершенстве: <br>
<FORM name="form1">
<input type="checkbox" name="lan" value=100>русский<br>
<input type="checkbox" name="lan" value=200>английский<br>
<input type="checkbox" name="lan" value=300>французский<br>
<input type="button" value=Вознагражденииие onClick="grant()"> <hr>
<input type="Text" size=50 name="ww" value=""><br>
<input type="reset" value="Отменить">
</FORM><hr>
</BODY>
</HTML>
```

Упражнение

Напишите сценарий обработки анкеты слушателя курсов. Пользователь может выбрать курс, его продолжительность, язык, на котором он готов работать с преподавателем, и форму отчетности. В зависимости от этих параметров определяется стоимость отдельного курса и стоимость всего обучения.

Списки

Если элементов много, то представление их с помощью флажков или переключателей увеличивает размер формы. В этом случае варианты выбора

могут быть представлены в окне браузера более компактно с помощью тега <select>. Тег имеет несколько параметров. Параметр name является обязательным. Для того чтобы установить число одновременно видимых элементов, следует задать параметр size=n. Когда n равно 1, то отображается ниспадающее меню или список выбора; при n>1 выводится список с n одновременно видимыми значениями. Если параметр size не задан, то по умолчанию принимается значение, равное 1. Указание параметра multiple означает, что из меню или списка можно выбрать несколько элементов. Элементы меню задаются внутри тега <select> с помощью тега <option>. Общий вид тега таков:

<option selected value=строка>

Параметр selected означает, что данный элемент списка считается выбранным по умолчанию. Параметр value содержит значение, которое передается, если данный элемент выбран из списка или меню.

Пример 1. Обработка анкеты переводчика

Напишем сценарий обработки анкеты переводчика. Сведения о тех языках, которыми владеет переводчик, требуется задать с помощью списка. Выбранные языки следует помещать в поле ввода многострочного текста.

Напомним, что событие change происходит в тот момент, когда значение элемента формы text, select или textarea изменилось и элемент потерял фокус. Будем обрабатывать анкету переводчика следующим образом. Параметр обработки события поместим в тег <select>. Как только выбран конкретный язык, т.е. произошло событие change, выполняется функция gr:

```
function gr(obj,m)
 { var r=100*(Number(((obj.elements[0])[m]).value)+1)
 s+=((obj.elements[0]) [m] ).text+"\r\n"
 obj.restext.value=s
 sum+=r
 obj.res.value=r}
```

Первый параметр - имя формы, второй - значение параметра value выбранного элемента. Второй оператор обеспечивает формирование строки всех выбранных пользователем языков. Третий оператор помещает вычисленное значение в текстовое поле. В результате выполнения четвертого оператора присваивания в переменной sum формируется значение, которое, затем при нажатии кнопки Сумма будет выведено в текстовое поле. Последний оператор помещает значение для выбранного языка в соответствующее поле формы. Полностью документ со сценарием и формами может быть описан так, как указано в листинге 1.

Листинг 1. Реакция на событие change в теге <select>

```
<HTML>
<HEAD>
<TITLE>Реакция на событие Change в теге select</TITLE>
<script language="JavaScript">
<!-- //
var s=""
var sum=0
function gr(obj,m)
{ var r=100*(Number(((obj.elements[0])[m]).value)+1)
s+=((obj.elements[0])[m]).text+"\r\n"
obj.restext.value=s
sum+=r
obj.res.value=r
}
//-->
</script>
</HEAD>
<BODY>
<FORM name="form1">
<H3>Анкета переводчика</H3>
<TABLE border=3 bgcolor=silver>
<TR><TH>Выбранный язык</TH><TH>Результат</TH></TR>
<TR>
<TD valign=top>
<select name="data" size=3 onChange="gr(form1,form1.data.value)">
<option value=0>русский
<option value=1>английский
<option value=2>французский
</select><P>
<input type="text" name="res" size=15>
</TD>
<TD><TEXTAREA name="restext" cols=15 rows=6>
</TEXTAREA><P>
<input type="button" value=Сумма onClick="form1.resgr.value=sum">
<input type="text" name="resgr" size=10>
</TD></TR></TABLE><p>
<input type="reset" value="Отменить" onClick="sum=0; s="">
</FORM>
</BODY>
</HTML>
```

Пример 2. Тест "Города и памятники"

Напишем сценарий обработки теста "Города и памятники". Названия

городов и памятников задаются с помощью списков. Пользователь выбирает в левом перечне название города, а в правом - памятник, расположенный в этом городе. После нажатия кнопки Результат в текстовое поле выводится количество правильных ответов.

В сценарии используются три глобальные переменные. Переменная q хранит последнее выбранное значение в левом столбце; переменная a - выбранное значение правого столбца; значение переменной sum содержит число правильных ответов. В двух списках для правильной пары "вопрос/ответ" совпадают соответствующие значения параметра value. Эти значения проверяются после выбора элемента списка правого столбца. Результат тестирования можно посмотреть, если нажать кнопку Результат.

Сценарий, реализующий простую обработку теста, представлен в листинге 2.

Листинг 2. Простая тестирующая программа

```
<HTML>
<HEAD>
<TITLE>Города и памятники.</TITLE>
<script>
<!--
var sum=0
var q
var a
function eq()
{ q=test.question.value
a=test.answer.value;
if (a==q) sum +=1
}
function result()
{ document.test.res.value=sum}
//-->
</script>
<BODY background="fon3.gif">
<h3 align=center>Города и памятники</h3>
<FORM name="test">
<TABLE border=3 align=center cellpadding=5
cellspacing=6 bgcolor= silver>
<TR><TH>Памятник</TH><TH>Находится в городе</TH>
<TR><TD>
<select size =7 name="question"
onChange="q=test.question.value">
<option value="mad">Музей Прадо<br>
<option value="ber" >Рейхстаг<br>
<option value="mil">Оперный театр Ла Скала<br>
```

```
<option value="ier">Стена Плача<br>
<option value="mek">Священный камень Кааб<br>
<option value="spb">Медный Всадник<br>
<option value="mos">Третьяковская галерея<br>
<option value="par">Триумфальная Арка<br>
<option value="new">Статуя Свободы<br>
<option value="lon">Тауэр<br>
</select>
</TD>
<TD>
<select size=7 name="answer" onChange="eq()">
<option value="spb">Санкт-петербург<br>
<option value="mos">Москва<br>
<option value="mek">Мекка<br>
<option value="ier">Иерусалим<br>
<option value="mil">Милан<br>
<option value="par">Париж<br>
<option value="mad">Мадрид<br>
<option value="lon">Лондон<br>
<option value="new">Нью-Йорк<br>
<option value="ber">Берлин<br>
</select>
</TD></TR>
</TABLE>
<CENTER><P>
<input type="button" value="Результат" onclick="result()"><br>
Количество правильных ответов
<input type="text" name="res" size="5"><P>
<input type="reset" value= "Обновить" onclick="sum=0">
</FORM>
</BODY>
</HTML>
```

Задания

1. Проверить примеры из лабораторной работы.

2. Напишите сценарий, который позволяет выбрать для таблицы и составляющих ее ячеек либо цвет фона, либо фоновое изображение, либо и то и другое. Предусмотрите возможность задания своего цвета фона для каждой ячейки.

3. Напишите сценарий, который позволяет посчитать стоимость предполагаемой покупки. Задается список продуктов, цена за единицу товара и количество экземпляров.

Практическая работа №15
Язык клиентских скприптов JSc

Цель работы*:* изучение возможностей языка JS, создание интерактивных форм при помощи клиентских скриптов.

Задание.

1. Создать файл index.html. Используя теги <HTML>, <HEAD>,<TITLE> и <BODY> описать общую структуру HTML документа. Создать форму анкетирования, как показано на рисунке.

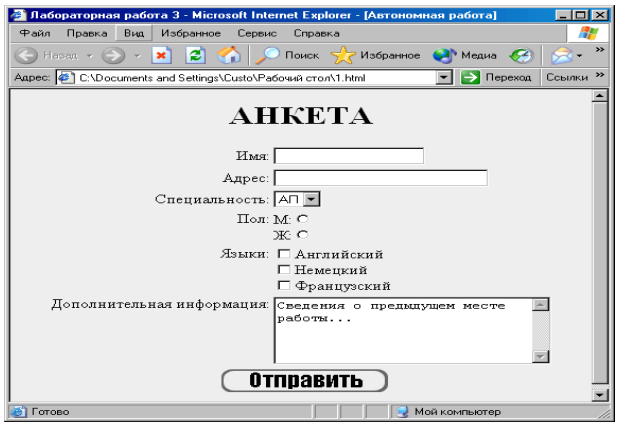


Рисунок. Страница с формой анкетирования

2. Вместо стандартной кнопки отправки формы использовать рисунок в формате GIF изображающий кнопку с эффектом RollOver (при наведении мышкой на кнопку она меняет цвет). Для создания кнопок использовать графический редактор Paint.

3. При помощи языка JS осуществить проверку содержимого основных полей формы (имя, адрес, специальность) при нажатии кнопки “Отправить”. При отсутствии данных хотя бы в одном из обязательных полей вывести соответствующее сообщение и прекратить обработку формы.

4. При правильном заполнении формы открыть страницу сайта, предназначенную для обработки данных формы.

5. Сделать выводы по работе.

ОГЛАВЛЕНИЕ

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Акимов С. В. Введение в Интернет-технологии. [Электронный ресурс]. URL: http://www.structuralist.narod.ru/it/internet/ internet.htm (дата обращения: 21.05.2017).
2. Андреев Д. PHP PDO. Краткое руководство. [Электронный ресурс]. URL: http://dandreev.com/blog/php-pdo-kratkoe-rukovodstvo/ (дата обращения: 12.06.2017).
3. Архитектура сайтов. // SEO бесплатный дистанционный курс. [Электронный ресурс]. URL: http://seo.abronova.com/ 62.php (дата обраще- ния: 21.03.2017).
4. Веллинг Л., Томсон Л. Разработка Web-приложений с помощью PHP и MySQL. – М.: Изд. дом «Вильямс», 2008. – 880 с.
5. Выскорко М. С. Регулярные выражения в Javascript. 24.11.2006. [Электронный ресурс]. URL: http://www.opennet.ru/ base/dev/ pcre_javascript.txt.html (дата обращения: 29.05.2017).
6. Гарретт Д. Д. Ajax: Новый подход к веб-приложениям /пер. А. Наконечного [Электронный ресурс]. URL: http://www.codenet.ru/webmast/ js/ajax/AJAX-New.php (дата обращения: 20.02.2017).
7. Доступ к базе данных в PHP /пер. С. Фастунова. 04.06.2012. // ruseller.com, Уроки. [Электронный ресурс]. URL: http://ruseller.com/ lessons.php?rub=37&id=1463 (дата обращения: 12.06.2017).
8. Дэвис Е. М., Филлипс Дж. А. Изучаем PHP и MySQL / пер. с англ. – СПб.: Символ-Плюс, 2008. – 448 с.
9. Евдокимов Н. В., Лебединский И. В. Раскрутка веб-сайта: практическое руководство по SEO 3.0. – М.: Изд-во «Вильямс», 2011. – 288 с.
10. Интернет. История. [Электронный ресурс]. URL: http://www.sunhome.ru/journal/12971 (дата обращения: 14.02.2017).
11. Кан М. Основы программирования на JavaScript. 2006 [Электронный ресурс]. URL: http://www.intuit.ru/department/ internet/jsbasics/ (дата обращения: 29.04.2017).
12. Кантор И. DOM: работа с HTML-страницей. [Электронный ресурс]. URL: http://javascript.ru/tutorial/dom (дата обращения: 17.05.2017).
13. Кантор И. Введение. DOM в примерах. // DOM: работа с HTML-страницей. [электронный ресурс]. URL: http://javascript.ru/tutorial/dom/ intro (дата обращения: 17.05.2017).
14. Кузнецов М., Симдянов И. PHP 5/6. – СПб.: БХВ-Петербург, 2010. – 1024 с.
15. Оптимизация сайта. Поисковая оптимизация сайта. SEO. [Электронный ресурс]. URL: http://webstudio2u.net/ru/site-optimization.html (дата обращения: 21.03.2012).
16. Поисковые системы и алгоритмы // SEO бесплатный дистанционный курс. [электронный ресурс]. URL: http://seo.abronova.com/22.php (дата

обращения: 21.03.2017).
17. Почему следует использовать PDO для доступа к базам данных? /пер. С. Фастунова. 23.06.2010. // ruseller.com, Уроки. [Электронный ресурс]. URL: http://ruseller.com/ lessons.php?rub=28&id=610 (дата обра- щения: 12.06.2017).
18. Савельев А. Технология, которая перевернет веб. // Компьютерра. 16.06.2005. [Электронный ресурс]. URL: http://www.computerra.ru/ hitech/39239/ (дата обращения: 29.02.2017).
19. Севостьянов И. О. Поисковая оптимизация: практическое руководство по продвижению сайта в Интернете. – СПб.: Питер, 2010. – 240 с.
20. Стандарт ECMA-262, 3-я редакция. [Электронный ресурс]. URL: http://javascript.ru/ecma (дата обращения: 03.05.2017).
21. Ташков П.А. Веб-мастеринг на 100%: HTML, CSS, JavaScript, PHP, CMS, AJAX, раскрутка. – СПб.: Питер, 2010. – 512 с.
22. Храмцов П. Б., Брик С. А., Русак А. М., Сурин А. И. Введение в JavaScript. 2003. [электронный ресурс]. URL: http://www.intuit.ru/ department/internet/js/ (дата обращения: 29.04.2012).
23. Шейда В. В. Защита информации в компьютерных сетях. Web уязви- мости: учебно-методическое пособие. – Томск: Томский гос. ун-т сис- тем управления и радиоэлектроники, 2007. – 68 с.
24. Эффекты перехода между страницами в HTML. [Электронный ресурс]. URL: http://perkoka.ru/article/10/664.html (дата обращения: 21.03.2017).
25. Apache friends – XAMPP. [Электронный ресурс]. URL: http://www.apachefriends.org/ru/xampp.html (дата обращения: 29.05.2017).
26. Harish Kamath. Регулярные выражения в JavaScript. 07.10.2007 / пер. В. Черкасова [Электронный ресурс]. URL: http://netsago.org/ru/docs/1/6/ (дата обращения: 25.05.2017).
27. PHP: Безопасность – Manual. [Электронный ресурс]. URL: http://www.php.net/manual/ru/security.php (дата обращения: 25.05.2017).
28. TIOBE Software: TIOBE index. [Электронный ресурс]. URL: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html (дата обращения: 29.02.2017).
29. XML-формат файла Sitemap. [электронный ресурс]. URL: http://www.sitemaps.org/ru/protocol.html#location (дата обращения: 25.03.2017).

Оглавление