УДК 004.439

РАЗРАБОТКА ЯЗЫКА РАЗМЕТКИ

А.К. Демидов

Предлагается язык разметки, обеспечивающий основные возможности форматирования и взаимодействие с CSS и JavaScript.

Ключевые слова: язык разметки.

Языки разметки. Информация, вводимая пользователем в CMS или на форуме, кроме текстовой части, может включать элементы форматирования (текст выделен жирным или мелким шрифтом), логическую структуру (текст является заголовком или элементом списка) и связи с другой информацией (фото или ссылка на другой сайт). Единственным способом внедрения этой информации во вводимый текст является язык разметки – набор символов или последовательностей, вставляемых в текст для передачи информации о его формате или строении. Язык разметки не определяет способ визуального отображения документа, что позволяет менять его отображение в зависимости от конкретного устройства вывода (монитор компьютера, мобильный телефон или принтер).

Несмотря на стремление разработчиков языков разметки упростить работу по добавлению разметки в текст, все языки требуют от пользователя запоминания необходимых синтаксических конструкций, ключевых слов и последовательностей. Однако при создании сложных документов могут потребоваться десятки различных элементов языка разметки и любые ошибки в использовании синтаксических конструкций или неправильная вложенность тегов могут привести некорректному форматированию текста. Чтобы пользователю не приходилось запоминать все ключевые слова и конструкции, на многих сайтах используют специальный редактор, позволяющий выделить часть текста и указать нужный вид форматирования этой части текста с помощью панели инструментов – все необходимые элементы языка разметки будут добавлены в текст и обеспечена их правильная вложенность. Главным недостатком такого редактора является то, что результат форматирования можно увидеть или только по специальному запросу на предпросмотр или после сохранения изменений на сайте. Для пользователя предпочтительнее использование WYSIWYG-редакторов, позволяющих изменять разметку, манипулируя визуальным отображением документа. Подобные редакторы существуют для HTML-разметки, но могут быть реализованы для других языков разметки. Если в тексте присутствуют элементы, форматирование которых производится автоматически (например, листинги с нумерацией строк, подсветка синтаксиса в программе, математические формулы), то для изменения таких элементов в WYSIWYG-редакторе должны быть предусмотрены специализированные мини-редакторы, на лету изменяющие форматирование элемента в соответствии с введенной информацией. Аналогичные проблемы возникают при использовании виджетов JQuery, формируемым на основе стандартных HTML-элементов. Редактор может быть использован для непосредственного ввода информации на сайте. Если же информация загружается из внешних источников, то при подготовке этой информации для загрузки снова возникает проблема её редактирования. При ориентировании только на WYSIWYG в каждое приложение, в котором будет готовиться информация, потребуется добавить инструменты для её редактирования и просмотра. Использование языков разметки минимизирует затраты на разработку приложений, но усложняет работу пользователям. Уровень знаний и квалификация пользователей должны быть тем выше, чем сложнее язык разметки и чем больше в нём правил и ограничений. Выделяют следующие основные разновидности языков разметки.

HTML (XML). Наиболее часто используемый язык, так как является стандартным для web-документов. Существует множество программ для WYSIWYG редактирования. Текстовые редакторы и офисные приложения могут экспортировать документы в этот формат, хотя при этом часто получается излишне перегруженный тегами и атрибутами документ. Прямое редактирование осложняется как наличием многочисленных тегов (более 100), требующих при использовании определенного порядка и вложенно-

сти, так и большой чувствительностью результата отображения к ошибкам разметки, которые трудно обнаруживать. При использовании языка HTML для загрузки информации необходима проверка текста на корректность (ошибка может привести к искажению отображения не только этой информации, но и всей страницы) и безопасности (внедрение в текст JavaScript дает возможность получить доступ к привилегиям и данным других пользователей, в т.ч. администратора). Основой для HTML является язык XML, поэтому все рассмотренные проблемы относятся к XML и к другим основанным на XML языкам разметки (DocBook, TEI, MathML и т.п.). Примером документа является:

<h1>Заголовок</h1>
Выделение текста курсивом и жирным
Ссылка

BB-коды. Были придуманы с целью предоставить более простой, безопасный и ограниченный по сравнению с HTML способ форматирования сообщений. BB-коды были созданы на основе HTML-тегов путем замены <> на [], и имеют те же ограничения по по применению, что и соответствующие HTML-теги. Преобразование текста в HTML производится специальной программой, проверяющей корректность использования тегов. Неверные теги и теги, для которых не нашлось соответствия, не преобразуются, и это упрощает поиск ошибок. Меньшее количество тегов (обычно не более 10–20) снижает требования к знаниям пользователя, но сужает возможности по форматированию информации. Также среди тегов отсутствуют теги для выделения логической структуры текста (например, нет тегов для заголовков). Близость к HTML позволяет использовать WYSIWYG-редактор, который после редактирования преобразует HTML-текст в текст с использованием BB-кодов. Чаще всего BB-коды используются на форумах, но зачастую каждый движок форума использует свой набор тегов. Наиболее распространены только 10 тегов. Примером документа является:

[center][size=5]Заголовок[/size][/center]
Выделение текста [i]курсивом[/i] и [b]жирным[/b]
[url=http://link.to/info.html]Ссылка[/url]

HAML в первую очередь используется как язык для описания шаблонов страниц, но может рассматриваться и как язык разметки полностью эквивалентный HTML/XML. Но наглядность структуры документа за счет обязательных отступов, отсутствие закрывающихся тегов значительно уменьшает вероятность ошибки по сравнению с HTML. Все теги HTML полностью доступны в HAML, но WYSYWYG-редакторов при этом не существует, поэтому работа с документами требует высокой квалификации пользователей. Примерами документа являются:

%h1 Заголовок %p Выделение текста %i курсивом	%strong жирным %a{:href=>"http://link.to/info.html"} Ссылка

TeX (LaTeX) используются в основном для форматирования научных статей, включают в себя специальные средства для выделения структуры документов, математических тестов, создания оглавлений и индексов. Сложность языка и требования к квалификации пользователя выше, чем у языка HTML. Аналогичный синтаксис имеет язык разметки Texinfo, используемый для генерации документов в проекте GNU (изменены названия тегов и вместо символа \ используется символ @). На научных форумах часто имеется возможность вставлять формулы в формате TeX, а для форматирования остальной части текста используются BB-коды или Wiki-разметка. Примером документа является:

\section{Заголовок}
Выделение текста \empf{курсивом} и \textbf{жирным}
\href{http://link.to/info.html}{Ссылка}

Wiki-разметка и др. При разработке языков Wiki-разметки, Texile, MarkDown, reStructuredText их авторы ставили цель упростить форматирование текста, повысить наглядность, некоторым образом приблизить к WYSIWYG без применения графических редакторов. Wiki-разметка, предложенная авторами MediaWiki и используемая, например, в Википедии, покрывает лишь небольшую часть необходимого форматирования текста, а для остальных элементов форматирования предлагается либо напрямую использовать теги HTML, либо вводятся HTML-подобные теги. Это можно рассматривать как неудачу в поставленной цели. Упущения в элементах форматирования породили большое количество несовместимых между собой вариантов языков Wiki-разметки, созданных для конкретных движков сайтов. В 2006–2008 годах была сделана попытка выработать стандарт для Wiki-разметки [1], который повысил наглядность языка разметки, но некоторые важные виды форматирования документов также были упущены. Кроме недостаточных возможностей, можно отметить необоснованные ограничения по использованию некоторых элементов форматирования внутри заголовков или ссылок и по строгой вложенности элементов, заимствованные из HTML. Язык Textile имеет те же проблемы, что и Wiki-разметка, хотя для расширенного форматирования абзацев и заголовков можно не прибегать к HTML. Наиболее близкий к результату отображения имеет документ с разметкой reStructuredText, а также его упрощенной разновидности MarkDown, но это достигается за счет большей работы пользователя (особенно при создании таблиц). Возможности форматирования при этом немногим больше чем в Wiki-разметке, в которой этот недостаток компенсируется использованием HTML. Примером документа является:

= Заголовок =
Выделение текста "курсивом" и '''жирным'''
[http://link.to/info.html Ссылка]

Требования, предъявляемые к языку разметки. Для создания сайтов с возможностью размещения новостей, статей, учебных материалов, тестов, с поддержкой форума или комментирования необходимы следующие элементы форматирования:

– определение логической структуры документа: заголовки разделов и подразделов, оглавление, блоки с дополнительной информацией;

– информационные единицы: абзацы текста, списки, таблицы, рисунки, пояснения, сноски, исходные тексты программ, формулы, цитаты;

– отступы и выравнивание для абзацев и таблиц;

– связи с другими данными: внешние ссылки, внутренние ссылки и якоря;

– выделение текста: курсивный и жирный шрифт, подчеркивание, вычеркивание, цвет и размер шрифта;

– разделители: горизонтальная линия, разрывы строк, страниц.

Для обеспечения взаимодействия с CSS и JavaScript необходимо иметь возможность указать класс и идентификатор элемента. Для ввода спецсимволов и смайлов можно использовать HTML-коды этих символов (&имя; или &#номер;) или похожие на них последовательности символов, например, `...` `(c)` `:-)`. Ввод элементов форматирования должен занимать минимальное время у пользователя, а внешний вид элементов форматирования – указывать на результат отображения форматированного текста. В этом случае наиболее простым вариантом для тегов форматирования является последовательности из 1, 2 или 3 повторений какого-либо символа. При этом последовательности из трех повторений используются только для определения информационных элементов, а последовательности из двух повторений – внутри информационных элементов. Одиночные символы используют либо когда эти символы практически не встречаются в обычном тексте, либо в начале строки, так как строка обычно начинается со слова, а не со знака пунктуации. Исключение составляют символы `-` `"` `'` `(` `$` и `.` как часть `...` За основу такого языка разметки можно взять Creole и DokuWiki [2].

Синтаксис предлагаемого языка разметки. Информационные элементы, занимающие несколько строк (абзацы, таблицы, списки) и не имеющие специального завершающего тега, разделяют одной или более пустой строкой. В остальных случаях пустая строка необязательна. Открывающие и завершающие теги блочных элементов обязательно должны начинаться с первой позиции, иначе они рассматриваются как обычные символы. Исключение: тег `[[[` в списках и таблицах. Для каждого тега можно указать класс элемента и/или идентификатор, добавив сразу после открывающего тега элемента форматирования #*идентификатор* или `.`*класс*. Идентификатор и

имя класса могут содержать только латинские буквы, цифры, символы _ и − и должны начинаться с буквы. Можно указать несколько классов у одного элемента форматирования, разделяя их символом . (точка).

Логическая структура документа. Заголовки разделов и подразделов начинаются с одного и более символов = в зависимости от уровня подраздела (не более 6). Между тегом и заголовком желательно, но не обязательно ставить пробел. Если в следующей строке указано такое же количество символов =, то строка рассматривается как продолжение заголовка в предыдущей строке. Для каждого подраздела автоматически создается якорь с именем подраздела.

Оглавление помещается в документ с помощью тега @@@. Оглавление создается на основе заголовков подразделов. После тега указывают число от 1 до 6 – ограничение на уровень заголовков, помещаемых в оглавление. Для вставки блока используются теги [[[и]]]. После открывающего тега можно указать ширину блока в пикселах или процентах и размещение блока с помощью символов > (справа) или < (слева) или ! (по центру, без обтекания). По умолчанию ширина блока равна 100 %, а размещение блока идет по центру. Допускается несколько уровней вложенности блоков. Для замечаний к тексту (ответов к задаче, подсказок к решению), выводимых только по запросу, используются тег +++. После отрывающего тега указывается метка, используемая в качестве имени кнопки для раскрытия дополнительной информации. Закрывающий тег не имеет метки.

Информационные единицы. Абзацами считаются последовательности строк, которые оканчиваются пустой строкой или открывающим/закрывающим тегом информационного элемента. В начале абзаца можно указать тег выравнивания текущего абзаца и указать умолчание для выравнивания следующих абзацев. Для задания выравнивания используются символы % (выравнивание влево с сохранением разбиения на строки, заданного пользователем), < (выравнивание влево), < (выравнивание вправо), : (выравнивание по ширине), ! (выравнивание по центру). С помощью двойного повторения тега выравнивания указывается выравнивание для последующих абзацев. По умолчанию с этой же целью применяется тег %. После тегов < и : указывают число от 0 до 9 – отступ для первой строки абзаца в символах. По умолчанию отступ для первой строки равен 0. Между тегом выравнивания и текстом желательно, но не обязательно ставить пробел. При установке умолчания для абзаца указание класса задает класс для следующих абзацев без тега выравнивания, но указание идентификатора относится только к текущему абзацу. Класс и идентификатор для абзаца можно указывать без тега выравнивания. Для списков используется теги * (обычный список) и # (нумерованный список). После тега обязательно должен быть пробел или указание класса и/или идентификатора. Количество про-

белов перед тегом определяет уровень вложенности списков, при этом учитывается относительное количество пробелов, а не абсолютное. На одном уровне должны быть одинаковые теги. Текст одного элемента списка может занимать несколько строк, текст выравнивается влево, а для разделения на строки используется явный тег \\). Для определения таблиц используется тег |. Первая строка таблицы является опциональной и должна начинаться с тега |, после него указывают класс и идентификатор, выравнивание и ширину таблицы в пикселах или процентах, а затем заголовок таблицы. Далее может следовать строка, содержащая ширину столбцов в пикселах или процентах и тег выравнивания (!, <, >) для тех ячеек, где выравнивание не указано явно. По умолчанию к ячейкам применяется выравнивание влево. Далее следуют строки таблицы. Ячейки таблицы разделяются символом |. Количество символов | в строках должно быть одинаковым. Если ячейка должна быть объединена с предыдущей ячейкой в строке, то символы | записываются без пропусков. Для объединения ячеек в нескольких строках в ячейке записывается последовательность символов :::. Заголовки строк и столбцов выделяются тегом =. Для выравнивания содержимого ячейки используются пробелы. Пробелы после текста означают выравнивание влево, пробелы перед тестом – вправо, пробелы до и после текста – по центру. Если пробелов нет, используется выравнивание, указанное в строке описание ширины столбцов. Для цитирования (в том числе на форуме) используется тег """. Автор и источник цитаты может быть указан после открывающегося или после закрывающего тега. Для выделения цитат внутри текста используется тег "". Исходный код может быть добавлен с помощью тега '''. В исходном коде не обрабатываются любые теги и HTML-коды. После открывающего тега можно указать имя языка программирования, если необходимо выделение резервированных слов, и заголовок. Для формул используются теги ` (формула AsciiMath), $$ (строчная формула TeX), $$$ (блочная формула TeX). В формулах не обрабатываются любые теги и HTML-коды. Преобразование формул происходит на стороне клиента с помощью MathJax. С помощью тега ++ можно добавить сноску. При этом сначала указывается имя сноски, а после символа | – сама сноска. При повторном использовании сноски достаточно указать её имя. Для определения ссылок используются теги [[и]]. Между тегами сначала указывается ссылка, а после символа | – текст. Если текст не указывается, то в качестве текста используется ссылка. Внешние ссылки обязательно должны начинаться с названия протокола, например, http://. Внутристраничные ссылки начинаются с символа @. Для создания якоря используется символы @@. Перед ссылкой можно указать её разновидность и символ :. Такая упрощенная ссылка будет преобразована в правильную форму. Внешние ссылки, не

имеющие текста, можно не заключать в скобки, они будут распознаны автоматически в тексте по протоколу. Для вставки картинок используется теги {{ и }}. Между тегами сначала указывается ссылка на картинку, а после символа | – альтернативный текст для картинки. После открывающего тега можно указать выравнивание картинки (!, <, >) и размеры картинки в пикселах или процентах, разделяя их символом x. Если символ x не указан, единственный размер интерпретируется как ширина картинки. По умолчанию картинка вставляется в строку, без обтекания. Указывается только один из размеров.

Выделение текста. Для выделения текста используются теги ** (жирный), // (курсивный), -- (вычеркнутый), __ (подчеркнутый), '' (равноширинный). Теги действуют как переключатели, первое появление включает режим, второе — выключает. Вложенность тегов не важна. Действие всех тегов завершается в конце абзаца, списка или таблицы. Для изменения размера, цвета или гарнитуры шрифта используются теги ((и)). После тега ((можно написать класс и/или идентификатор. Далее можно указать имя гарнитуры, имя или номер цвета, размер шрифта в точках, процентах или с помощью имени (xxs, xs, s, m, l, xl, xxl). Теги ((и)) можно использовать как для постоянного переключения, так и для временного. В случае временного переключения после параметров указывается символ |, после него указывается выделяемый текст. Равноширинный текст используется автоматически для одиночных символов и HTML-кодов в апострофах.

Разделители и отступы. Для разделения текста горизонтальной линией используется тег ---, после него можно указать класс, идентификатор, выравнивание и ширину линии. Для принудительного разделения текста на страницы используется тег \\\. Тег ::: применяют для установки отступа слева. После него указывают отступ в символах. По умолчанию отступ устанавливается равным 0.

Прочие теги. Для внутренних комментариев в тексте используются теги {{{ и }}}. Пользователь, имеющий соответствующие права, может использовать эти теги для определения JavaScript и CSS. Экранирующий символ ~ используется в том случае, когда тег включен в обычную последовательность символов. Для вывода самого символа ~ используется последовательность символов ~~. Для определения неразрывных пробелов нужно указать символ ~ перед пробелом. Пример документа:

= Заголовок
Выделение текста //курсивом// и **жирным**
[[http://link.to/info.html|Ссылка]]

Реализация обработчика языка разметки. При преобразовании текста с использованием разметки для просмотра в браузере необходимо сначала найти все теги и последовательности, разделяющие абзацы в этом

тексте, а затем выполнить их замену на некоторые последовательности HTML-тегов, зависящие от предыдущих обработанных тегов. Как показано в [3], использование регулярных выражений для поиска тегов в тексте, доступные во всех современных языках программирования, приводит к экспоненциальному росту времени обработки для сложных регулярных выражений. Это связано с тем, что практически все реализации регулярных выражений используют бэктрекинг вместо недетермированных конечных автоматов. Наиболее эффективным вариантом с точки зрения производительности для создания обработчика языка разметки является использование генераторов лексических анализаторов (Flex для C/C+, JLex для Java, PHPLex для PHP), которые по набору правил создают конечный автомат. Дополнительная информация по разработке предлагаемого языка разметки содержится в [4].

Библиографический список

1. Creole 1.0. A common wiki markup. – URL: http://www.wikicreole.org/.
2. URL: http://www.dokuwiki.org/wiki:syntax.
3. Cox, R. Regular Expression Matching Can Be Simple And Fast. – URL: http://swtch.com/~rsc/regexp/regexp1.html.
4. URL: http://ipc.susu.ac.ru/24719.html.