The RenderX Tribune

RenderX News

2006-05-15

XEP 4.6 with AFP output released. RenderX has released XEP 4.6, a new version of its XSL processor and accompanying tools. The new version introduces AFP backend (available with a special license), improved linebreaking algorithm conformant to the Unicode Standard Annex #14, and a new implementation of XSL 1.1 change bars. XEPwin 2.0 has also been updated and includes XEP 4.6.

Wikipedia

Tail recursion w In computer science, tail recursion (or tail-end recursion) is a special case of recursion that can be easily transformed into an iteration. Such a transformation is possible if the recursive call is the last thing that happens in a function. Replacing recursion with iteration, either manually or automatically, can drastically decrease the amount of stack space used and improve efficiency. This technique is commonly used with functional pro-

...continued on page 2

Lisp, the Programming Language

"Lisp is a programmable programming language." -John Foderaro

What is Lisp?

LISP is an acronym for LISt Processing. Its development history has often been associated with symbolic processing and with both computer and human languages. A heterogeneous list data type has always been built into the language in order to efficiently deal with arbitrary and changing models.

From this basis, a number of features have come to be expected from all members of the Lisp family of languages:

- top-level in which code can be run
- decoupled syntax from semantics
- compiled / interpreted
- language / libraries / environment
- object-oriented / procedural
- dynamic / static
- strong / weak typing
- memory mangagement
- macros
- effective use of computational resources
- 3GL/4GL/5GL

Lisp has evolved into a family of languages. The two major dialects in use today are Common Lisp and Scheme. This web site deals mostly with Common Lisp.

A paper titled Common Lisp: Myths and Legends tries to confront and dispel some of the myths and mistaken impressions associated with Lisp. The paper promotes Xanalys's Lisp products, but most of the arguments apply to other Common Lisp implementations as well.

Kent Pitman's More Than Just Words: Lambda, the Ultimate Political Party argues that Lisp is better defined as its community than as its various specifications.

Lisp may be combined with other languages to produce wide variety of applications.

There have been a number of comparisons between Lisp and other languages.

Lisp History

Lisp was invented by John McCarthy in the late 1950's as a formalism for reasoning about the use of recursion equations as a model for computation. Of computer languages still in widespread use today, only FORTRAN is older.

Lisp has evolved with the field of Computer Science, always putting the best ideas from the field into practical use. In 1994, Common Lisp became the first ANSI standard to incorporate object oriented programming.

The early development of Lisp is described by McCarthy in "History of Lisp", 1978.

...continued on page 2

W3C News

2006-07-14

Semantic Web Activity Grows to Include GRDDL, Deployment Working Groups W3C is pleased to announce the renewal of the Semantic Web Activity. "W3C continues to support the advancement of universal sharing and automatic processing of data in the World Wide Web," said Ivan Herman (W3C). Semantic Web technologies allow data to be shared and reused across applications, enterprises, and communities. The W3C Advisory Committee approved the continuing work in RDF data access, rules interchange, and health care and life sciences. Three new groups are chartered for work on Semantic Web deployment, extracting RDF from XML (e.g., to process microformats), and education and outreach. Join W3C and visit the Semantic Web home page.

2006-07-17

W3C Names Yves Lafon Web Services Activity LeadW3C has named Yves Lafon to the position of Web Services Activity Lead. The Web Services Activity includes Working Groups for semantic anno-

...continued on page 2

AI Koans

A novice was trying to fix a broken lisp machine by turning the power off and on. Knight, seeing what the student was doing spoke sternly- "You can not fix a machine by just power-cycling it with no understanding of what is going wrong."

Knight turned the machine off and on.

The machine worked.

One day a student came to Moon and said, "I understand how to make a better garbage collector. We must keep a reference count of the pointers to each cons."

Moon patiently told the student the following story-

"One day a student came to Moon and said, "I understand how to make a better garbage collector...

In the days when Sussman was a novice Minsky once came to him as he sat hacking at the PDP-6. "What are you doing?", asked Minsky.

"I am training a randomly wired neural net to play Tic-Tac-Toe."

"Why is the net wired randomly?", asked Minsky.

"I do not want it to have any preconceptions of how to play"

Minsky shut his eyes,

"Why do you close your eyes?", Sussman asked his teacher.

...continued on page 4

Wikipedia *(continued from page 1)*

gramming languages where the declarative approach and explicit handling of state promote the use of recursive functions that rapidly fill the call stack.

Iteration

The word iteration is sometimes used in everyday English with a meaning virtually identical to repetition.

Iteration in mathematics may refer to the process of iterating a function, or to the techniques used in iterative methods for solving numerical problems.

Iteration in computing is the repetition of a process within a computer program. It can be used both as a general term, synonymous with repetition, and to describe a specific form of repetition with a mutable state.

When used in the first sense, recursion is an example of iteration, but typically using a recursive notation, which is typically not the case for iteration.

However, when used in the second (more restricted) sense, iteration describes the style of programming used in imperative programming languages. This contrasts with recursion, which has a more declarative approach.

Call stack

In computer science, a call stack is a special stack which stores information about the active subroutines of a computer program. (The active subroutines are those which have been called but have not yet completed execution by returning.) This kind of stack is also known as an execution stack, control stack, or function stack, and is often abbreviated to just "the stack".

A call stack is often used for several related purposes, but the main reason for having one is to keep track of the point to which each active subroutine should return control when it finishes executing. If, for example, a subroutine DrawSquare calls a subroutine DrawLine from four different places, the code of DrawLine must have a way of knowing which place to return to. This is typically done by code for each call within DrawSquare putting the address of the instruction after the particular call statement (the "return address") into the call stack.

...continued on page 3

Lisp, the Programming Language *(continued from page 1)*

Two of the major developers of the language since then, Richard Gabriel and Guy Steele, presented "The Evolution of Lisp" at the 1993 ACM History of Programming Languages conference.

Kent Pitman and Brad Miller have compiled a brief on-line version of the history of Lisp from ANSI documents.

Herbert Stoyan began his study of the history of Lisp in the early 1970s, while in East Germany. He carried out this study by writing letters to everyone whose name he could find in the documents as well as by getting every book and report he could. In 1979 he applied to emigrate from East Germany, was arrested, spent six months in prison and then was permitted to emigrate to West Germany. When he got out of East Germany, he visited M.I.T. and examined every document he could find relevant to the history of Lisp and also interviewed everyone he could. He is now Professor of Computer Science at the University of Erlangen-Nuremberg in Erlangen, Germany, where his group works on AI. His history of Lisp is also on-line.

What is Lisp Good For?

ANSI Common Lisp is a mature, thoughtfully conceived, highly portable, industrial-strength programming language which serious application developers worldwide have come to count on for:

— Highly customizable "quick and dirty" utilities for doing everyday things. (i.e. as the most powerful scripting language available).
— Large, complicated, mission critical applications which would be impossible to develop in any other language.
— Fast prototyping and Rapid Application Development (RAD).
— Continuous-availability applications, especially those that require functionality changes after initial deployment.

It has had wide success in business process software, engineering, document processing, hypermedia (including the WWW), mathematics, graphics and animation, artificial intelligence and natural language processing.

It is sometimes used to define every aspect of an application, sometimes just the internal processing engine (i.e. the guts, without the User Inteface), or sometimes just the user interface. Sometimes it is used as the "glue" to interactively develop Graphical User Interfaces and sometimes to define graphics/windowing systems themselves.

It is often used to provide interactive command languages, macro or scripting languages, and extension languages embedded within commercial systems.

It is widely used to programatically generate other, static, stand-alone applications, in the same or different language. It is also used to generate "mobile code".

Xanalys, a major commercial Lisp vendor, has placed an excellent paper titled Common Lisp: Myths and Legends on their site. From the introduction:

"This year, 1998, Lisp is celebrating its 40th year supporting the world's most complex applications. And Lisp has grown a lot in that time. So if you or someone you know harbors fears or concerns about Lisp because of something learned in a class or muttered by a friend 20 or 30 years ago, it's time to take a fresh look. In this paper we're going to survey what Lisp is today."

This site lists a sampling of commercial uses, listed by industry, and reasearch uses.

This site also gives a description of Lisp and its features, available tools written in Lisp, and a comparison with other languages.

...continued on page 3

W3C News *(continued from page 1)*

tations, addressing, choreography, description and policy as well as XML protocol and databinding. Yves joined W3C in 1995 to work on the experimental browser Arena. He led development of Jigsaw, W3C's Java based server, and served as Activity Lead for the Protocols Activity and the XML Protocol Activity. and as Team Contact for the XML Protocol Working Group, the XML Schema Patterns for Databinding Working Group and the Web Services Choreography Working Group. W3C wishes to thank Hugo Haas who previously led the Activity. Read more about W3C. (Photo credit: Coralie Mercier. News archive)

2006-07-12

mobileOK Scheme: Working Draft The Mobile Web Best Practices Working Group has released the First Public Working Draft of the W3C mobileOK Scheme 1.0. mobileOK labels indicate that content and its delivery pass tests based on the Mobile Web Best Practices and are designed to create an effective user experience. Read about the W3C Mobile Web Initiative, a joint effort by authoring tool vendors, content

...continued on page 3

Lisp, the Programming Language *(continued from page 2)*

Using Lisp with Other Languages

Summary: Lisp programs can be combined with programs written in other languages, to form an application that runs within a Lisp top-level or from some other controlling program.

"We must all hang together, or most assuredly, we will all hang separately." -Ben Franklin

In general, a Lisp implementation provides a Lisp top-level that allows code to be loaded into it using the function LOAD. Any programs loaded in this way and called from top-level are under the control of the Lisp top-level.

Exactly what kind of files can be loaded with LOAD is implementation dependent. In some implementations, LOAD can handle .o, .so, or .dll files produced by external operating system-specific compilers. In other implementations, a new top-level must be constructed which includes the external program.

However, there is nothing in the definition of Common Lisp that specifies that all programs must be run from the Lisp top-level. For example, Lisp could be provided as a library of utilities that are linked to some other controlling program. There are some implementations that provide such a library, while others have the ability to effectively create one for the programmer, based on the Lisp source files for a specific application.

In any case, LOAD and/or the Lisp compiler must be informed about functions or other operations to be defined in a different language. Exactly how this is done is implementation and operating-system specific. The various mechanisms are collectively known as foreign-function interfaces. Usually, these involve some specification of:

— what other language is being used for each foreign utility to be used (so that the compiler can use the correct calling convention).
— the arguments and return values types for the function.

Some possible complications occur if:

— The Lisp implementation uses utilities that are inherently incompatible with utilities used in a program to be linked with Lisp. This is rare.
— Lisp data is to be passed directly to external utilities and kept in variables within the external program. In some cases this, can interfere with Lisp memory management because:
  — The memory manager of some Lisp implementations have the ability to move data within memory (to obtain better performance), so Lisp data pointed to by foreign programs could move, resulting in pointers that are no longer valid.
  — The Lisp memory manager can keep track of which data is being used within the Lisp system, but may not be able to keep track of data that is being used by external programs.

  Most Lisp implementations provide special facilities for dealing with these issues.

An alternative mechanism for using applications involving multiple languages is to use an object-based inter-object protocol. This is done in such systems as CORBA Object Request Brokers (ORBs), ILU, and OLE, and these will work with many Lisp implementations.

In an inter-object protocol, a programmer specifies information about the interface between program modules, and indicates where the modules are to be found. The various modules might be:

— within the same process (i.e. executable application) as Lisp -- in which case the foreign function interface discussion, above, is relevant.

...continued on page 4

Wikipedia *(continued from page 2)*

Since the call stack is organized as a stack, the caller pushes the return address onto the stack, and the called subroutine, when it finishes, pops the return address off the call stack (and transfers control to that address). If a called subroutine calls on to yet another subroutine, it will push its return address onto the call stack, and so on, with the information stacking up and unstacking as the program dictates. If the pushing consumes all of the space allocated for the call stack, an error called a stack overflow occurs. Adding a subroutine's entry to the call stack is sometimes called winding; conversely, removing entries is unwinding.

There is exactly one call stack associated with a running program (or more accurately, with each task of a process). Since there is only one in this important context, it can be referred to as the stack (implicitly, "of the task").

In high-level programming languages, the specifics of the call stack are usually hidden from the programmer. They are given access only to the list of functions, and not the memory on the stack itself. Most assembly languages on the other hand, require programmers to be involved with manipulating the stack. The actual details of the stack in a programming language depend upon the compiler, operating system, and the available instruction set.

Common Lisp

Common Lisp, commonly abbreviated CL, is a dialect of the Lisp programming language, standardised by ANSI X3.226-1994. Developed to standardize the divergent variants of Lisp which predated it, it is not an implementation but rather a language specification. Several implementations of the Common Lisp standard are available, including commercial products and open source software.

Common Lisp is a general-purpose programming language, in contrast to Lisp variants such as Emacs Lisp and AutoLISP which are embedded extension languages in particular products. Unlike many earlier Lisps, Common Lisp (like Scheme) uses lexical variable scope.

Common Lisp is a multi-paradigm programming language that:

...continued on page 4

W3C News *(continued from page 2)*

providers, handset manufacturers, browser vendors and mobile operators.

2006-07-11

Advance Notice: Workshop on Requirements for XSL-FO 2.0W3C plans a Workshop on Gathering Requirements for Extensible Stylesheet Language (XSL) 2.0 on 18 October in Heidelberg, Germany, hosted by Heidelberger Druckmaschinen AG. The Workshop will be held in conjunction with a symposium on Web Printing at the same location. Participants will discuss the requirements, features and design of a future version of the formatting part of the Extensible Stylesheet Language also called XSL-FO. A Call for Participation for this Workshop is expected in August. Read about W3C Workshops and the XML Activity.

2006-07-11

XML Query and XPath Data Model Updated The XML Query and XSL Working Groups have released an updated Candidate Recommendation of XQuery 1.0 and XPath 2.0 Data Model (XDM). Both XSLT 2 and XQuery use XPath expressions and operate

...continued on page 4

AI Koans *(continued from page 1)*

"So that the room will be empty."

At that momment, Sussman was enlightened.

A student, in hopes of understanding the Lambda-nature, came to Greenblatt. As they spoke a Multics system hacker walked by. "Is it true", asked the student, "that PL-1 has many of the same data types as Lisp". Almost before the student had finshed his question, Greenblatt shouted, "FOO!", and hit the student with a stick.

A disciple of another sect once came to Drescher as he was eating his morning meal. "I would like to give you this personality test", said the outsider,"because I want you to be happy." Drescher took the paper that was offered him and put it into the toaster- "I wish the toaster to be happy too".

A cocky novice once said to Stallman: "I can guess why the editor is called Emacs, but why is the justifier called Bolio?". Stallman replied forcefully, "Names are but names, `Emac & Bolio's` is the name of a confectionary shop in Boston-town. Nei-

Lisp, the Programming Language *(continued from page 3)*

— in different processes on the same machine.
— on different machines connected over a network.

The issues involved with such inter-object protocols are largely the same for applications written in any combination of languages, including those that are partly written in Lisp.

LispWorks

Xanalys's LispWorks (R) for the Windows (R) operating system ("LWW") runs on Windows NT, 95 and 98. On Unix platforms LispWorks runs on Sun Sparc and clones (SunOS and Solaris), IBM RS/6000 (AIX), DEC Alpha (OSF/1), HP PA (HP-UX), and SGI (IRIX). Liquid CL (formerly Lucid CL) runs on the same range of unix platforms with the exception of DEC Alpha.

All editions implement the ANSI Common Lisp standard and come with native CLOS/MOP, incremental compiler, interpreter, dynamic loader, and tools for inspection, profiling, disassembly, stepping and tracing. Language extensions include: full multithreading, foreign language interface, defsystem, support for internationalization through Unicode, programmer-extensible streams, TCP socket streams, object finalization, weak vectors and hash-tables, LALR parser generator, CAPI portable GUI toolkit.

All editions support the Common LispWorks IDE, which provides a smooth and comfortable workflow, allowing you to incrementally write, test, and extend your software while it is running. Features include: interactive listener; debugger; object inspector; browsers for classes, generic functions, compilation conditions; profiler; integrated extensible editor; build system manager; source code location and cross-referencing tool; complete online documentation.

The Professional Edition of LWW includes everything you need for commercial Common Lisp software development and application delivery. CLIM 2.0 is included to further increase program portability. Applications developed with the Professional edition can be distributed free of charge.

The Enterprise Edition of LWW extends the Professional Edition, providing portable distributed computing through CORBA, database access through object-oriented SQL/ODBC libraries, expert system programming through KnowledgeWorks and an embedded Prolog compiler. On Unix the SQL interface is part of the LispWorks product and libraries supporting product delivery, CORBA, CLIM and KnowledgeWorks may be purchased separately.

On the other hand the Personal Edition of LWW is intended for personal and educational Lisp programming. As a contribution to the Common Lisp community, Xanalys is making the Personal Edition of LispWorks for Windows available free of charge from its Web site. While the Personal Edition includes the full Common Lisp compiler and development environment, it does limit program size and duration and it does not support application delivery.

Wikipedia *(continued from page 3)*

— Supports programming techniques such as imperative, functional and object-oriented programming.
— Is dynamically typed, but with optional type declarations that can improve efficiency.
— Is extensible through standard features such as Lisp macros (compile-time code rearrangement accomplished by the program itself) and reader macros (extension of syntax to give special meaning to characters reserved for users for this purpose).

Functional programming

Functional programming is a programming paradigm that conceives computation as the evaluation of mathematical functions and avoids state and mutable data. Functional programming emphasizes the application of functions, in contrast to imperative programming, which emphasizes changes in state and the execution of sequential commands.

Functional programming is defined more by a set of common concerns and themes than any list of distinctions from other paradigms. Often considered important are higher-

W3C News *(continued from page 3)*

on XDM instances such as documents and databases. The group also released an updated Working Draft of the XQuery Update Facility which provides expressions to create, modify and delete nodes. Visit the XML home page.

2006-07-10

Rule Interchange Format Use Cases and Requirements Updated The Rule Interchange Format (RIF) Working Group has published an updated Working Draft of RIF Use Cases and Requirements. Synthesized from nearly fifty use cases, the document specifies use cases and requirements for a format that allows rules to be translated between rule languages and thus transferred between rule systems. The group invites comments through 8 September. Visit the Semantic Web home page.

2006-07-05

Last Call: XHTML Modularization 1.1 The HTML Working Group has released a Last Call Working Draft of XHTML Modularization 1.1. This modularization allows the subsets and extensions to XHTML needed for emerging platforms. This document is based on Modularization of XHTML in XML Schema and the Modularization of XHTML W3C Rec-