CIS 189

Week 8 Team Project

This week, you will work in teams of two to create an XML file that stores data, plus an HTML + JavaScript web page that transforms that data for viewing in a web browser.

Your project must meet the following requirements:

1. Each team will create:

a) An XML file containing appropriate data for the projects (20 points)  
Team member names must be a comment on the second or third line of the file.  
Your XML structure will be graded for clarity: are related elements properly nested?   
Do the child/sibling/parent relationships reflect the nature of the data?  
Good XML structure should be easy to read. XML elements generally correspond to rows in tables or records in a database. If you have to think deeply about how things relate to make connections, you may want to try another approach.  
All XML files must have at least 20 of the major data element:  
20 temperature readings, 20 students, etc.

b) An HTML + JavaScript file that formats the XML suitable for display and demonstrates

 Load an XML file into JavaScript (5 points)

 Select an XML element by name (5 points)

 Iterate through/processing child nodes (5 points)

 Filter by node type (5 points)

 Use node values from the XML DOM in HTML code (5 points)

c) A CSS file for styling (10 points)

2. Team member names must be a comment on the second or third line of each source file to receive credit for that part of the lab

3. You are welcome to find example HTML, CSS, and JavaScript code on-line. If you include nontrivial code from someone else's example, please include the URL in a comment in the program.  
*You will still receive full credit for your project if you cite your source.*  
Code included without citation can have serious academic consequences.

4. The HTML, XML, and related files must work when loaded through Visual Studio on the instructor's computer! (10 points)

5~~. Your team will demonstrate your completed XML-and-other files to the class  
on the first class meeting of week 9. (20 points)~~

6. Extra effort and creativity will be rewarded. (15 points)

7. When the project is complete, please zip all related files into a single folder and turn in the zip file to the professor on-line. The project must work on the professor’s computer using only the project files submitted through Canvas.

8. ~~All files you want to use for your presentation, including slides or similar, must be submitted through Canvas by end-of-day before the presentation.~~

Sample Project topics

1. The XML file contains a list of Celsius temperatures from a weather station. The data for each readingincludes a city name, a date, a time, and a decimal temperature. Display using a <table> in the web browser.

2. The XML file contains a list of students, courses completed, and grades earned. Use a CSS file to display the information in an attractive, professional way in a web browser.

3. The XML file contains a list of quiz questions and answers. Display the questions and answers in an attractive manner using CSS. Use JavaScript to display a timer or similar.

4. The XML file contains a list of employees, their employee IDs, and their salaries. Display a list of employees who make more than one million dollars a year.

5. The XML file contains a list of record albums, artist names, year of publication, and any other information you deem suitable. Use a web browser to display the list of albums using an unsorted list (<ul>) HTML element.

6. Create an interactive HTML+JS quiz program that uses an XML file as the source of its quiz questions.

You are welcome to suggest other projects, but they must be approved by the instructor within one hour of the start of team projects.

HOW TO APPROACH THIS PROJECT

You have several class sessions to complete this! You will put together something rough and improve it as time goes on.

Day 1: Team up, form project plan, start initial coding

◦ Form your team

◦ Choose from the instructor-provided list of topics

◦ Sketch out the big picture of what is required and how development can be split up

◦ Create a small XML file that demonstrates the structure you'll use for the final project

▪ In most teams, one person will be in charge of populating the XML file, the other will be in charge of initial HTML/CSS/JS development

◦ Make sure to start with something small that works and build up!

Development days: Coordinate with your team

◦ Populate XML file with data as required

◦ Add features to HTML/CSS/JS to make the information display more dynamic or attractive

◦ Confirm that the program works on the instructor computer - you are welcome to test before the due date!

Last day: create a writeup

* A writeup template is provided at the end of this document
* Take one or more screenshots of your project displayed in a web browser
* Copy-and-paste the code from your HTML/JS, CSS, and XML files

~~Last day: Show off!~~

~~◦ The file must be submitted to the instructor by end-of-day the day before the presentations. Files submitted after this time~~ *~~will not be accepted for presentation~~*~~.~~

~~◦ The instructor will load your team's program on the instructor workstation~~

~~◦ Take a few minutes to tell the class:~~

~~▪ Who is on your team~~

~~▪ What you've created~~

~~▪ Show off the web page~~

~~▪ Point out one or two neat things you've done in the code~~

~~ Can be as simple as a for loop or as complex as object-level coding in JavaScript using XML data~~

~~◦ Plan to talk to the class for about 5 minutes. It may be easier to present if the whole team comes up, but it is acceptable appoint a spokesperson for your team.~~

CHECKLIST FOR SUBMISSION

o The team has created related XML, plus CSS and HTML/JS files

o The XML file has at least 20 of the major data element (such as 20 students, 20 music albums, 20 employees ...)

o Team members have their names in the comments of *all* source code files, 2nd or 3rd line

o All required technologies are used: XML, HTML, JS, CSS

o A test confirms all required files are being sent to the instructor

o The files work on Firefox on the instructor workstation

o Submission is in by end-of-day on the due date

o If there are presentations this quarter, the submission is due end-of-day the night before presentations

RUBRIC

XML data file: 20

JavaScript: 25 points total

Load XML: 5

Select by name: 5

Child nodes: 5

Filter by type: 5

Insert nodeValue: 5

CSS: 10

Works on Instructor’s Computer: 10

Presentation (or writeup for quarters without presentations): 20

Extra effort: 15

SAMPLE XML FILE COMMENT

<!-- Project created for CIS 189 by Janet Weiss and Joe Ortega -->

SAMPLE PROJECT WRITEUP

Team member names: Jamal Brown and Seo Yeon Nugyen

Screenshot(s):

Graphical user interface, text

Description automatically generated

Copy-and-paste your HTML/JS, CSS, and XML code:

<HTML>

. . . etc.

head { }

. . . etc.

<?Xml version=”1.0”>

. . . etc.

WRITEUP TEMPLATE

Team member names: Jordan McNair and Jude Mai

Screenshot(s):



Copy-and-paste your HTML/JS, CSS, and XML code:

(HTML/JS, CSS, and XML code goes here.

Paste the code as text. Screenshots will not be accepted for credit.)

<?xml version="1.0" encoding="utf-8"?>

<?xml-stylesheet href="TeamStyle.css"?>

<!-- Jordan McNair & Jude Mai -->

<WebPage>

<entry>

<language>C#</language>

<description>C# (C-Sharp) is a programming language developed by Microsoft that runs on the .NET Framework. C# is used to develop web apps, desktop apps, mobile apps, games and much more.</description>

<paradigm>Multi-paradigm: structured, imperative, object-oriented, event-driven, task-driven, functional, generic, reflective, concurrent</paradigm>

<family>C</family>

<designer>Anders Hejlsberg</designer>

<initialRelease>2001</initialRelease>

</entry>

<entry>

<language>JavaScript</language>

<description>JavaScript is the Programming Language for the Web. JavaScript can update and change both HTML and CSS. JavaScript can calculate, manipulate and validate data.</description>

<paradigm>Multi-paradigm: event-driven, functional, imperative, procedural, object-oriented programming</paradigm>

<family>JavaScript</family>

<designer>Brendan Eich &amp; others</designer>

<initialRelease>December 4th 1995</initialRelease>

</entry>

<entry>

<language>SQL</language>

<description>SQL is a standard language for storing, manipulating and retrieving data in databases. Our SQL tutorial will teach you how to use SQL in: MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres, and other database systems.</description>

<paradigm>Declarative</paradigm>

<family>Query language</family>

<designer>Donald D. Chamberling, Raymond F. Boyce</designer>

<initialRelease>April 24th 1989</initialRelease>

</entry>

<entry>

<language>Swift</language>

<description>Swift is the new programming language for iOS and OS X development. It borrows from languages such as C, Objective-C, and other modern programming languages.</description>

<paradigm>Multi-paradigm: protocol-oriented, object-oriented, functional, imperative, block structured, declarative, concurrent</paradigm>

<family>C</family>

<designer>Chris Lattner, Doug Gregor, John McCall, Ted Kremenek, Joe Groff, and Apple Inc.</designer>

<initialRelease>June 2nd 2014</initialRelease>

</entry>

<entry>

<language>PHP</language>

<description>PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages. PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.</description>

<paradigm>Multi-paradigm: imperative, functional, object-oriented, procedural, reflective</paradigm>

<family>C</family>

<designer>Rasmus Lerdorf</designer>

<initialRelease>1995</initialRelease>

</entry>

<entry>

<language>Ruby</language>

<description>Ruby is an interpreted, high-level, general-purpose programming language which supports multiple programming paradigms. It was designed with an emphasis on programming productivity and simplicity.</description>

<paradigm>Multi-paradigm: functional, imperative, object-oriented, reflective</paradigm>

<family>Scripting language</family>

<designer>Yukihiro Matsumoto</designer>

<initialRelease>December 25th 1996</initialRelease>

</entry>

<entry>

<language>Java</language>

<description>Developers use Java to construct applications in laptops, data centres, game consoles, scientific supercomputers, cell phones, and other devices. Java is the world's third most popular programming language, after Python and C.</description>

<paradigm>Multi-paradigm: generic, object-oriented (class-based), functional, imperative, reflective, concurrent</paradigm>

<family>C</family>

<designer>James Gosling</designer>

<initialRelease>January 23rd 1996</initialRelease>

</entry>

<entry>

<language>Go</language>

<description>Go (also called Golang or Go language) is an open source programming language used for general purpose. Go was developed by Google engineers to create dependable and efficient software. Most similarly modeled after C, Go is statically typed and explicit.</description>

<paradigm>Multi-paradigm: concurrent imperative, object-oriented</paradigm>

<family>C</family>

<designer>Robert Griesemer, Rob Pike, and Ken Thompson</designer>

<initialRelease>March 2012</initialRelease>

</entry>

<entry>

<language>TypeScript</language>

<description>TypeScript extends JavaScript and improves the developer experience. It enables developers to add type safety to their projects. Moreover, TypeScript provides various other features, like interfaces, type aliases, abstract classes, function overloading, tuple, generics, etc.</description>

<paradigm>Multi-paradigm: functional, generic, imperative, object-oriented</paradigm>

<family>JavaScript</family>

<designer>Microsoft Corporation</designer>

<initialRelease>October 1st 2012</initialRelease>

</entry>

<entry>

<language>Rust</language>

<description>What is Rust programming used for? Rust is a high-performance, statically-typed multi-paradigm programming language. With the main focus on safety and performance. This language helps developers create robust and secure applications.</description>

<paradigm>Multi-paradigm: concurrent, functional, generic, imperative, structured</paradigm>

<family>ML-family (with C-family sytnax)</family>

<designer>Graydon Hoare</designer>

<initialRelease>July 7th 2010</initialRelease>

</entry>

<entry>

<language>COBOL</language>

<description>A compiled English-like computer programming language designed for business use.</description>

<paradigm> Multi-paradigm: procedural, imperative, object-oriented, generic</paradigm>

<family>COBOL</family>

<designer>Howard Bromberg, Norman Discount, Vernon Reeves, Jean E. Sammet, William Selden, Gertrude Tierney, with indirect influence from Grace Hopper</designer>

<initialRelease>1959</initialRelease>

</entry>

<entry>

<language>F#</language>

<description>A functional-first, general purpose, strongly typed, multi-paradigm programming language that encompasses functional, imperative, and object-oriented programming methods. It is most often used as a cross-platform Common Language Infrastructure (CLI) language on .NET, but can also generate JavaScript[8] and graphics processing unit (GPU) code.</description>

<paradigm> Multi-paradigm: functional, imperative, object-oriented, agent-oriented, metaprogramming, reflective, concurrent</paradigm>

<family>ML</family>

<designer>Don Syme, Microsoft Research</designer>

<initialRelease>2005</initialRelease>

</entry>

<entry>

<language>Python</language>

<description>Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.</description>

<paradigm>Multi-paradigm: object-oriented, procedural (imperative), functional, structured, reflective</paradigm>

<family>ABC</family>

<designer>Guido van Rossum</designer>

<initialRelease>February 20th 1991</initialRelease>

</entry>

<entry>

<language>C</language>

<description>A general-purpose computer programming language. It was created in the 1970s by Dennis Ritchie, and remains very widely used and influential. By design, C's features cleanly reflect the capabilities of the targeted CPUs. It has found lasting use in operating systems, device drivers, protocol stacks, though decreasingly[6] for application software. C is commonly used on computer architectures that range from the largest supercomputers to the smallest microcontrollers and embedded systems.</description>

<paradigm> Multi-paradigm: imperative (procedural), structured</paradigm>

<family>C</family>

<designer>Dennis Ritchie</designer>

<initialRelease>1972</initialRelease>

</entry>

<entry>

<language>C++</language>

<description>A high-level general-purpose programming language created by Danish computer scientist Bjarne Stroustrup as an extension of the C programming language, or "C with Classes". The language has expanded significantly over time, and modern C++ now has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation. It is almost always implemented as a compiled language, and many vendors provide C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, Embarcadero, Oracle, and IBM, so it is available on many platforms.</description>

<paradigm> Multi-paradigm: procedural, imperative, functional, object-oriented, generic, modular</paradigm>

<family>C</family>

<designer>Bjarne Stroustrup</designer>

<initialRelease>1985</initialRelease>

</entry>

<entry>

<language>Turing</language>

<description> a high-level, general-purpose programming language developed in 1982 by Ric Holt and James Cordy, at University of Toronto in Ontario, Canada. It was designed in order to help students taking their first computer science course learn how to code. Turing is a descendant of Pascal, Euclid, and SP/k that features a clean syntax and precise machine-independent semantics.</description>

<paradigm>Multi-paradigm: object-oriented, procedural</paradigm>

<family>Pascal</family>

<designer>Ric Holt and James Cordy</designer>

<initialRelease>1982</initialRelease>

</entry>

<entry>

<language>Pascal</language>

<description>An imperative and procedural programming language, designed by Niklaus Wirth as a small, efficient language intended to encourage good programming practices using structured programming and data structuring. It is named in honour of the French mathematician, philosopher and physicist Blaise Pascal.</description>

<paradigm> Imperative, structured</paradigm>

<family>Wirth Pascal</family>

<designer>Niklaus Wirth</designer>

<initialRelease>1970</initialRelease>

</entry>

<entry>

<language>Euclid</language>

<description>An imperative programming language for writing verifiable programs. It was designed by Butler Lampson and associates at the Xerox PARC lab in the mid-1970s.</description>

<paradigm> Multi-paradigm: structured, imperative, functional</paradigm>

<family>Pascal</family>

<designer>Butler Lampson, Xerox PARC</designer>

<initialRelease>1970s</initialRelease>

</entry>

<entry>

<language>A++</language>

<description>A++ stands for abstraction plus reference plus synthesis which is used as a name for the minimalistic programming language that is built on ARS-based programming. ARS-based programming is used as a name for programming which consists mainly of applying patterns derived from ARS to programming in any language. ARS is an abstraction from the Lambda Calculus, taking its three basic operations, and giving them a more general meaning, thus providing a foundation for the three major programming paradigms functional programming, object-oriented programming and imperative programming.</description>

<paradigm>Multi-paradigm: functional, object-oriented, imperative, logic</paradigm>

<family>ARS</family>

<designer>Georg P. Loczewski, Britain Hamm</designer>

<initialRelease>1996-2002</initialRelease>

</entry>

<entry>

<language>F\*</language>

<description>A functional programming language inspired by ML and aimed at program verification. Its type system includes dependent types, monadic effects, and refinement types.</description>

<paradigm>Multi-paradigm: functional, imperative</paradigm>

<family>ML</family>

<designer>Microsoft Research, Inria</designer>

<initialRelease>Unknown</initialRelease>

</entry>

</WebPage>

<?xml version="1.0" encoding="utf-8"?>

<?xml-stylesheet href="TeamStyle.css"?>

<!-- Jordan McNair & Jude Mai -->

<WebPage>

<entry>

<language>C#</language>

<description>C# (C-Sharp) is a programming language developed by Microsoft that runs on the .NET Framework. C# is used to develop web apps, desktop apps, mobile apps, games and much more.</description>

<paradigm>Multi-paradigm: structured, imperative, object-oriented, event-driven, task-driven, functional, generic, reflective, concurrent</paradigm>

<family>C</family>

<designer>Anders Hejlsberg</designer>

<initialRelease>2001</initialRelease>

</entry>

<entry>

<language>JavaScript</language>

<description>JavaScript is the Programming Language for the Web. JavaScript can update and change both HTML and CSS. JavaScript can calculate, manipulate and validate data.</description>

<paradigm>Multi-paradigm: event-driven, functional, imperative, procedural, object-oriented programming</paradigm>

<family>JavaScript</family>

<designer>Brendan Eich &amp; others</designer>

<initialRelease>December 4th 1995</initialRelease>

</entry>

<entry>

<language>SQL</language>

<description>SQL is a standard language for storing, manipulating and retrieving data in databases. Our SQL tutorial will teach you how to use SQL in: MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres, and other database systems.</description>

<paradigm>Declarative</paradigm>

<family>Query language</family>

<designer>Donald D. Chamberling, Raymond F. Boyce</designer>

<initialRelease>April 24th 1989</initialRelease>

</entry>

<entry>

<language>Swift</language>

<description>Swift is the new programming language for iOS and OS X development. It borrows from languages such as C, Objective-C, and other modern programming languages.</description>

<paradigm>Multi-paradigm: protocol-oriented, object-oriented, functional, imperative, block structured, declarative, concurrent</paradigm>

<family>C</family>

<designer>Chris Lattner, Doug Gregor, John McCall, Ted Kremenek, Joe Groff, and Apple Inc.</designer>

<initialRelease>June 2nd 2014</initialRelease>

</entry>

<entry>

<language>PHP</language>

<description>PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages. PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.</description>

<paradigm>Multi-paradigm: imperative, functional, object-oriented, procedural, reflective</paradigm>

<family>C</family>

<designer>Rasmus Lerdorf</designer>

<initialRelease>1995</initialRelease>

</entry>

<entry>

<language>Ruby</language>

<description>Ruby is an interpreted, high-level, general-purpose programming language which supports multiple programming paradigms. It was designed with an emphasis on programming productivity and simplicity.</description>

<paradigm>Multi-paradigm: functional, imperative, object-oriented, reflective</paradigm>

<family>Scripting language</family>

<designer>Yukihiro Matsumoto</designer>

<initialRelease>December 25th 1996</initialRelease>

</entry>

<entry>

<language>Java</language>

<description>Developers use Java to construct applications in laptops, data centres, game consoles, scientific supercomputers, cell phones, and other devices. Java is the world's third most popular programming language, after Python and C.</description>

<paradigm>Multi-paradigm: generic, object-oriented (class-based), functional, imperative, reflective, concurrent</paradigm>

<family>C</family>

<designer>James Gosling</designer>

<initialRelease>January 23rd 1996</initialRelease>

</entry>

<entry>

<language>Go</language>

<description>Go (also called Golang or Go language) is an open source programming language used for general purpose. Go was developed by Google engineers to create dependable and efficient software. Most similarly modeled after C, Go is statically typed and explicit.</description>

<paradigm>Multi-paradigm: concurrent imperative, object-oriented</paradigm>

<family>C</family>

<designer>Robert Griesemer, Rob Pike, and Ken Thompson</designer>

<initialRelease>March 2012</initialRelease>

</entry>

<entry>

<language>TypeScript</language>

<description>TypeScript extends JavaScript and improves the developer experience. It enables developers to add type safety to their projects. Moreover, TypeScript provides various other features, like interfaces, type aliases, abstract classes, function overloading, tuple, generics, etc.</description>

<paradigm>Multi-paradigm: functional, generic, imperative, object-oriented</paradigm>

<family>JavaScript</family>

<designer>Microsoft Corporation</designer>

<initialRelease>October 1st 2012</initialRelease>

</entry>

<entry>

<language>Rust</language>

<description>What is Rust programming used for? Rust is a high-performance, statically-typed multi-paradigm programming language. With the main focus on safety and performance. This language helps developers create robust and secure applications.</description>

<paradigm>Multi-paradigm: concurrent, functional, generic, imperative, structured</paradigm>

<family>ML-family (with C-family sytnax)</family>

<designer>Graydon Hoare</designer>

<initialRelease>July 7th 2010</initialRelease>

</entry>

<entry>

<language>COBOL</language>

<description>A compiled English-like computer programming language designed for business use.</description>

<paradigm> Multi-paradigm: procedural, imperative, object-oriented, generic</paradigm>

<family>COBOL</family>

<designer>Howard Bromberg, Norman Discount, Vernon Reeves, Jean E. Sammet, William Selden, Gertrude Tierney, with indirect influence from Grace Hopper</designer>

<initialRelease>1959</initialRelease>

</entry>

<entry>

<language>F#</language>

<description>A functional-first, general purpose, strongly typed, multi-paradigm programming language that encompasses functional, imperative, and object-oriented programming methods. It is most often used as a cross-platform Common Language Infrastructure (CLI) language on .NET, but can also generate JavaScript[8] and graphics processing unit (GPU) code.</description>

<paradigm> Multi-paradigm: functional, imperative, object-oriented, agent-oriented, metaprogramming, reflective, concurrent</paradigm>

<family>ML</family>

<designer>Don Syme, Microsoft Research</designer>

<initialRelease>2005</initialRelease>

</entry>

<entry>

<language>Python</language>

<description>Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.</description>

<paradigm>Multi-paradigm: object-oriented, procedural (imperative), functional, structured, reflective</paradigm>

<family>ABC</family>

<designer>Guido van Rossum</designer>

<initialRelease>February 20th 1991</initialRelease>

</entry>

<entry>

<language>C</language>

<description>A general-purpose computer programming language. It was created in the 1970s by Dennis Ritchie, and remains very widely used and influential. By design, C's features cleanly reflect the capabilities of the targeted CPUs. It has found lasting use in operating systems, device drivers, protocol stacks, though decreasingly[6] for application software. C is commonly used on computer architectures that range from the largest supercomputers to the smallest microcontrollers and embedded systems.</description>

<paradigm> Multi-paradigm: imperative (procedural), structured</paradigm>

<family>C</family>

<designer>Dennis Ritchie</designer>

<initialRelease>1972</initialRelease>

</entry>

<entry>

<language>C++</language>

<description>A high-level general-purpose programming language created by Danish computer scientist Bjarne Stroustrup as an extension of the C programming language, or "C with Classes". The language has expanded significantly over time, and modern C++ now has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation. It is almost always implemented as a compiled language, and many vendors provide C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, Embarcadero, Oracle, and IBM, so it is available on many platforms.</description>

<paradigm> Multi-paradigm: procedural, imperative, functional, object-oriented, generic, modular</paradigm>

<family>C</family>

<designer>Bjarne Stroustrup</designer>

<initialRelease>1985</initialRelease>

</entry>

<entry>

<language>Turing</language>

<description> a high-level, general-purpose programming language developed in 1982 by Ric Holt and James Cordy, at University of Toronto in Ontario, Canada. It was designed in order to help students taking their first computer science course learn how to code. Turing is a descendant of Pascal, Euclid, and SP/k that features a clean syntax and precise machine-independent semantics.</description>

<paradigm>Multi-paradigm: object-oriented, procedural</paradigm>

<family>Pascal</family>

<designer>Ric Holt and James Cordy</designer>

<initialRelease>1982</initialRelease>

</entry>

<entry>

<language>Pascal</language>

<description>An imperative and procedural programming language, designed by Niklaus Wirth as a small, efficient language intended to encourage good programming practices using structured programming and data structuring. It is named in honour of the French mathematician, philosopher and physicist Blaise Pascal.</description>

<paradigm> Imperative, structured</paradigm>

<family>Wirth Pascal</family>

<designer>Niklaus Wirth</designer>

<initialRelease>1970</initialRelease>

</entry>

<entry>

<language>Euclid</language>

<description>An imperative programming language for writing verifiable programs. It was designed by Butler Lampson and associates at the Xerox PARC lab in the mid-1970s.</description>

<paradigm> Multi-paradigm: structured, imperative, functional</paradigm>

<family>Pascal</family>

<designer>Butler Lampson, Xerox PARC</designer>

<initialRelease>1970s</initialRelease>

</entry>

<entry>

<language>A++</language>

<description>A++ stands for abstraction plus reference plus synthesis which is used as a name for the minimalistic programming language that is built on ARS-based programming. ARS-based programming is used as a name for programming which consists mainly of applying patterns derived from ARS to programming in any language. ARS is an abstraction from the Lambda Calculus, taking its three basic operations, and giving them a more general meaning, thus providing a foundation for the three major programming paradigms functional programming, object-oriented programming and imperative programming.</description>

<paradigm>Multi-paradigm: functional, object-oriented, imperative, logic</paradigm>

<family>ARS</family>

<designer>Georg P. Loczewski, Britain Hamm</designer>

<initialRelease>1996-2002</initialRelease>

</entry>

<entry>

<language>F\*</language>

<description>A functional programming language inspired by ML and aimed at program verification. Its type system includes dependent types, monadic effects, and refinement types.</description>

<paradigm>Multi-paradigm: functional, imperative</paradigm>

<family>ML</family>

<designer>Microsoft Research, Inria</designer>

<initialRelease>Unknown</initialRelease>

</entry>

</WebPage>

<html>

<!-- Jordan McNair & Jude Mai // 11.2.2022 -->

<head>

<link href="TeamStyle.css" rel="stylesheet" />

</head>

<body>

<script>

/\*...SCRIPT LINEAR LOGIC...\*/

var fileName = "TeamTree.xml";

var recursionLevel = 0;

var savedFields = [];

var x = loadXMLDoc(fileName);

if (!x)

alert("XML file did not load.");

else {

traverse(x.documentElement);

}

/\* LOAD ENTRIES ON TO PAGE USING THE SAVED XML FIELDS\*/

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

var lang = savedFields[i];

i++;

var desc = savedFields[i];

i++;

var para = savedFields[i];

i++;

var fam = savedFields[i];

i++;

var desi = savedFields[i];

i++;

var ini = savedFields[i];

i++;

AddEntry(lang, desc, para, fam, desi, ini);

}

/\*...DEFINING THE FUNCTIONS...\*/

/\* FUNCTION TO TRAVERSE THROUGH CURRENT NODE \*/

function traverse(currentNode) {

if (currentNode.childNodes.length > 0) {

recursionLevel++;

for (var i = 0; i < currentNode.childNodes.length; i++) {

// outputNode(currentNode, recursionLevel)

var result = traverse(currentNode.childNodes[i]);

}

if (currentNode.nodeType == 1) {

savedFields.push(currentNode);

}

/\*AFTER AN ELEMENT IS CREATED FOR EACH XML FIELD IT IS ADDED TO THE ADD ENTRY FUNCTION WHICH PLACES IT WITHIN A DIV ELEMENT\*/

recursionLevel--;

}

}

/\* FUNCTION THAT IDENTIFIES SPACES \*/

function spaces(n) {

var myStr = "";

for (var i = 0; i < n; i++)

myStr += "&nbsp&nbsp&nbsp&nbsp";

return (myStr);

}

/\* FUNCTION THAT LOADS AN XML DOCUMENT \*/

function loadXMLDoc(filename) {

xhttp = new XMLHttpRequest();

xhttp.open("GET", filename, false);

xhttp.send();

return xhttp.responseXML;

}

/\* FUNCTION ADDS NEW ELEMENTS TO AN ENTRY DIV\*/

function AddEntry(lang,desc,para,fam,des,ini) {

// create a new entry

const newEntry = document.createElement("div");

lang.setAttribute("id", "h1");

desc.setAttribute("id", "h2");

para.setAttribute("id", "h3");

fam.setAttribute("id", "h4");

des.setAttribute("id", "h5");

ini.setAttribute("id", "h6");

// add the newly created element to the entry "div"

newEntry.appendChild(lang);

newEntry.appendChild(desc);

newEntry.appendChild(para);

newEntry.appendChild(fam);

newEntry.appendChild(des);

newEntry.appendChild(ini);

// add the newly created element and its content into the DOM

const currentElement = document.getElementById("div");

document.body.insertBefore(newEntry, currentElement);

}

</script>

</body>

</html>