Multitasking Algorithms in XForms

Coordination, Progress, Priority

Presented July 31, 2019 at

Balisage: The Markup Conference 2019

By John M. Boyer, Ph.D.

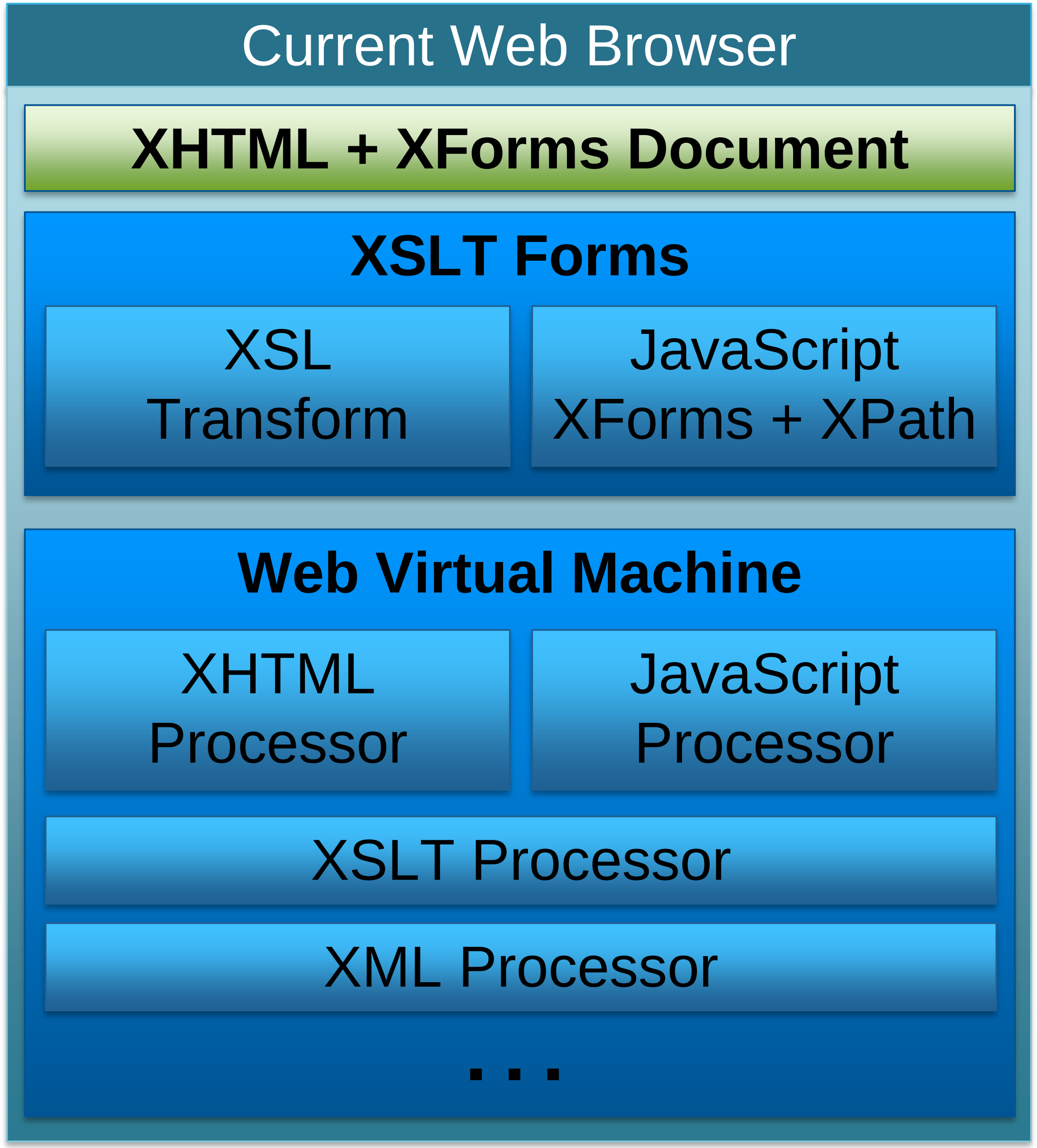
Overview

- Introduction
- Background on XForms Markup
  - Data representation and referencing
  - Declarative data and user interface dependencies
  - Imperative scripting for data processing use cases
- Building Data Processing Algorithms in XForms
  - Invoking data processing scripts
  - A selected algorithm to show the Turing-complete features
- Multitasking Techniques for XForms Actions
  - Coordination
  - Progress
  - Priority
- Conclusion

Introduction

- What is XForms?
  - A multiparadigm language for orchestrating user interaction with XML data in a document
  - Offers declarative, imperative, and even-driven language features that work together
  - Fits within XHTML, ODF, SVG, and other XML document formats
- Who made XForms?
  - A W3C working group and the world wide web community
  - The latest “standard” (full W3C Recommendation) is XForms 1.1
  - **This year is the 10 Year Anniversary of XForms 1.1!**
  - The working group created further specifications and extension features as an XForms community
  - The code in this paper is intended for XForms 1.1 or higher

Introduction: Processor Architecture



Current Web Browser

XHTML + XForms Document

XSLT Forms

XSL Transform

JavaScript XForms + XPath

Web Virtual Machine

XHTML Processor

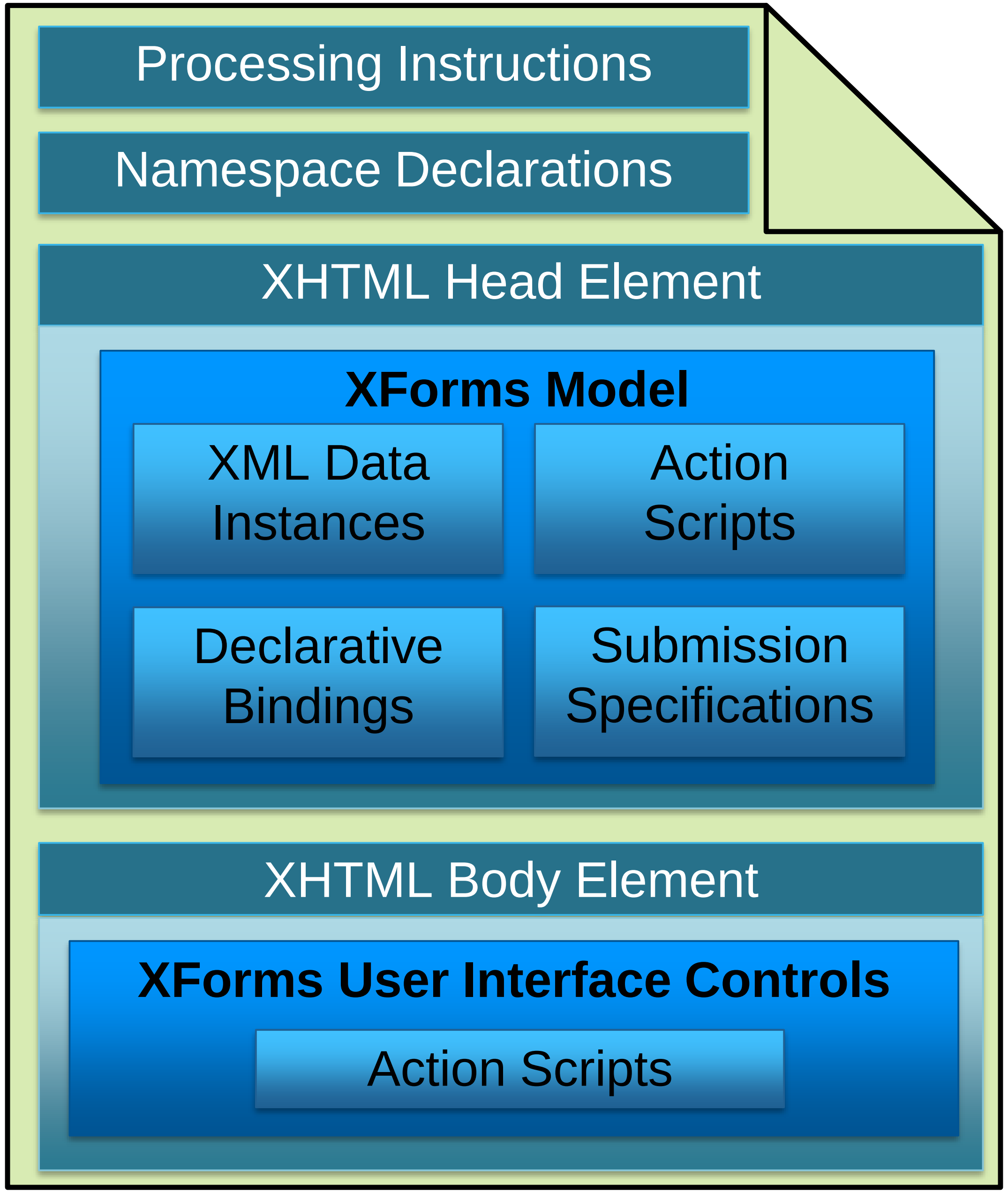
JavaScript Processor

XSLT Processor

XML Processor

. . .

Introduction: Document Architecture



Processing Instructions

Namespace Declarations

XHTML Head Element

XForms Model

XML Data Instances

Action Scripts

Declarative Bindings

Submission Specifications

XHTML Body Element

XForms User Interface Controls

Action Scripts

Background on XForms Markup

- Processing Instruction for XSLTForms
  - The first line of the XHTML

```
<?xml-stylesheet href="xsltforms/xsltforms.xsl" type="text/xsl"?>
```

- The XForms Model (in the XHTML head element)
  - In the XHTML head element
  - The ‘id’ and ‘xmlns’ attributes
  - Container for data instances, action scripts, etc.

```
<model xmlns="http://www.w3.org/2002/xforms" id="M">
   …
</model>
```

Background: Data Representation

- XForms data instances
  - In the <model> element, each is represented by a parsed XML tree
  - Loaded from external source, or internal inline declaration

```
<instance id="ORIGDATA" src="./Sort_Data_10.xml"/>
<instance id="DATA">
  <data xmlns="">
    <items>
      <item>
        <num>5</num>
        <name>Alice</name>
        <phone>333-1111</phone>
      </item>
      …
    </items>
  </data>
</instance>
```

Background: Data Referencing

- XPath Expressions
  - Instance data nodes are referenced with XPath
  - Separate instances referenced using ‘id’ attribute value in the *instance()* function that XForms adds to its XPath engine

    instance(‘DATA’)/items/item/name

  - An XPath can bind to multiple nodes that match an expression
  - The empty namespace (xmlns="") was used on the data within instances to simplify XPath references in the code

Background: Declarative Embellishment

- Data Dependencies
  - Content values of data nodes can be calculated by XPath expressions
  - Each XPath is an automatic declarative embellishment of other language operations, such as the behaviors of script commands

```
<bind nodeset="instance('VARS')/window/end"
        calculate="../start + ../size - 1"/>
```

- User Interface Dependencies
  - Input and output user interface control elements can bind to data nodes to show or allow editing of their values
  - UI bindings are declarative embellishments on any other language operations that change the bound data nodes

```
<output ref="instance('VARS')/window/end"/>
```

Background: Imperative Data Manipulators

- Setting the Text Content Value of a Data Node

```
<setvalue ref="instance('VARS')/window/start"
          value="choose(. > 1, . - ../size, .)"/>
```

- Deleting Nodes

```
<delete nodeset="instance('DATA')/items/item"/>
```

- Inserting Nodes into a Parent Node

```
<insert context="instance('DATA')/items"
        origin="instance('ORIGDATA')/items/item"/>
```

Background: Scripts as Event Handlers

- Using a container action that handles an XML event:

```
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:ev="http://www.w3.org/2001/xml-events">
  <head>
    …
    <model xmlns="http://www.w3.org/2002/xforms" id="M">
      <instance id="ORIGDATA" src="./Sort_Data_10.xml"/>
      …
      <action ev:event="xforms-model-construct-done">
        <delete nodeset="instance('DATA')/items/item"/>
        <insert context="instance('DATA')/items"
                origin="instance('ORIGDATA')/items/item"/>
      </action>
    </model>
  </head>
  …
</html>
```

Background: User Interface Controls

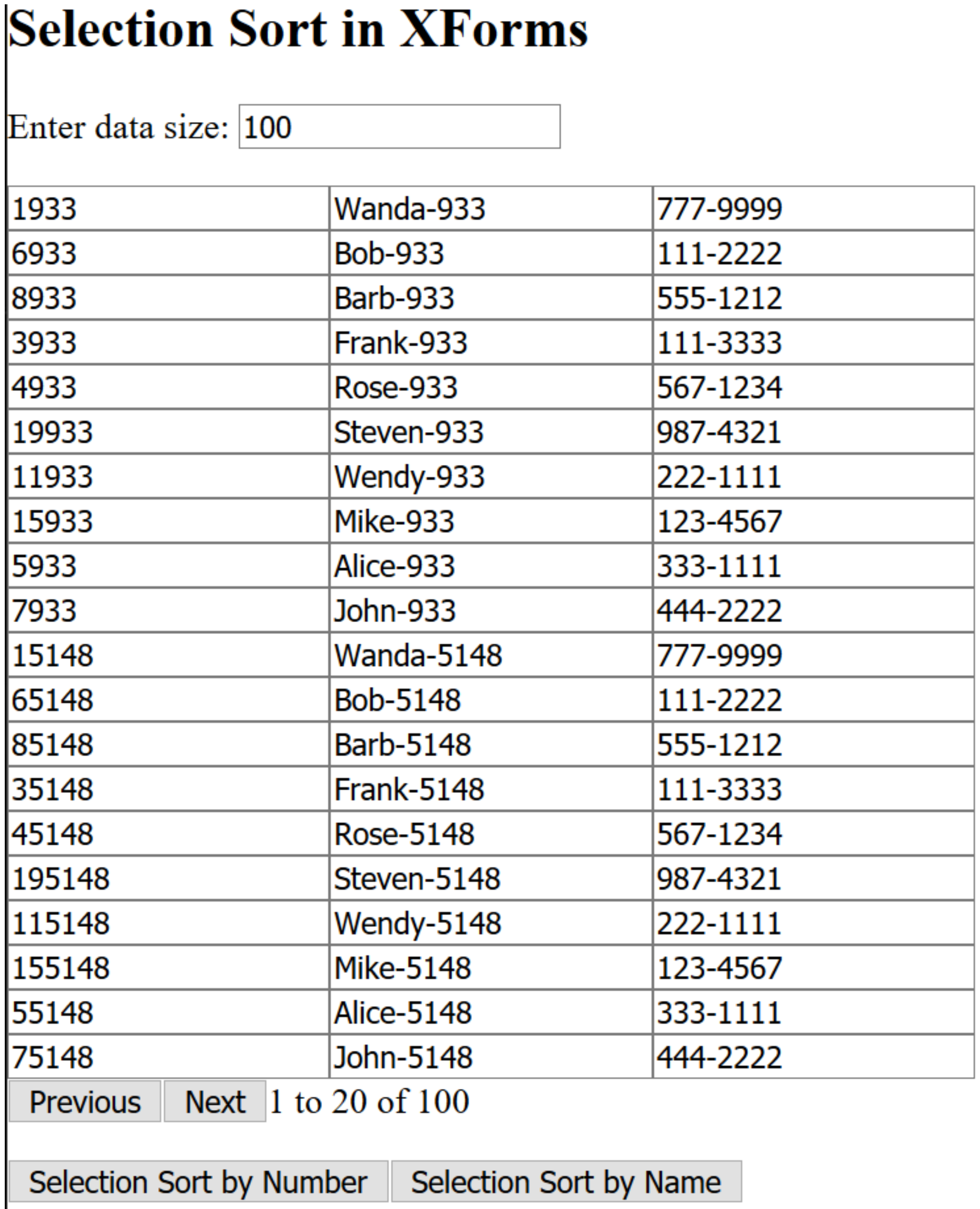
- Making a button control that performs a script

```
<trigger>
   <label>Previous</label>
   <action ev:event="DOMActivate">
        <setvalue ref="instance('VARS')/window/start"
                       value="choose(. > 1, . - ../size, .)"/>
   </action>
</trigger>
```

- Making the windowing table that takes user input:

```
<repeat nodeset="instance('DATA')/items/item[
                           position() >= instance('VARS')/window/start and
                           position() &lt;= instance('VARS')/window/end]">

            <input ref="num"><label/></input>

            <input ref="name"><label/></input>

            <input ref="phone"><label/></input>

</repeat>
```

Data Processing Algorithm in XForms: Demo



Selection Sort in XForms

Enter data size: 100

1933	Wanda-933	777-9999
6933	Bob-933	111-2222
8933	Barb-933	555-1212
3933	Frank-933	111-3333
4933	Rose-933	567-1234
19933	Steven-933	987-4321
11933	Wendy-933	222-1111
15933	Mike-933	123-4567
5933	Alice-933	333-1111
7933	John-933	444-2222
15148	Wanda-5148	777-9999
65148	Bob-5148	111-2222
85148	Barb-5148	555-1212
35148	Frank-5148	111-3333
45148	Rose-5148	567-1234
195148	Steven-5148	987-4321
115148	Wendy-5148	222-1111
155148	Mike-5148	123-4567
55148	Alice-5148	333-1111
75148	John-5148	444-2222

1 to 20 of 100

Data Processing Algorithms: Invocation

- Invoking a model script from the user interface with <*dispatch*>:

```
<html … >
   <head>
      <model xmlns="http://www.w3.org/2002/xforms" id="M">
         …
         <action ev:event="sel-sort-num">
            …
         </action>
      </model>
   </head>
   <body>
      …
      <trigger>
         <label>Selection Sort by Number</label>
         <dispatch ev:event="DOMActivate"
                   name="sel-sort-num" targetid="M"/>

      </trigger>
      …
   </body>
</html>
```

Data Processing Algorithms: Sort, I

- Make an instance for algorithm control variables
  - ‘items’ is used to take a copy of the item elements to sort
  - ‘sorted’ is used to grow the list of item elements in sorted order
  - ‘least’ is used to help find the element with the least key in ‘items’

```
<instance id="SVARS">
        <sort xmlns="">
           <items/>
           <sorted/>
           <least/>
        </sort>
</instance>
```

Data Processing Algorithms: Sort, II

- The outer logical level of the selection sort
  - Make a copy of the items to sort
  - Loop through to find the minimum and move it to the sorted list
  - Move the sorted list to the original location in instance data

```
<action ev:event="sel-sort-num">

   <insert context="instance('SVARS')/items"
           origin="instance('DATA')/items/item"/>

   <action while="instance('SVARS')/items/item">
      …
   </action>

   … <!-- Actions to move the sorted list to original location -->
</action>
```

Data Processing Algorithms: Sort, III

- The inner logical level of the selection sort
  - Initialize the ‘least’ to the first item’s key
  - Iterate to find the least
    - Use declarative infusion for numeric keys or an XForms loop for strings
    - Use declarative infusion to solve for multiple nodes having equal keys
  - Move the minimum item from the unsorted list to the sorted list

```
<action while="instance('SVARS')/items/item">

    <setvalue ref="instance('SVARS')/least" value="../items/item[1]/num"/>

    <setvalue ref="instance('SVARS')/least" value="min(../items/item/num)"/>

    <insert context="instance('SVARS')/sorted"
            nodeset="item" at="last()" position="after"
            origin="../items/item[num=instance('SVARS')/least]"/>

    <delete nodeset="instance('SVARS')/items/item[
                                    num=instance('SVARS')/least]"/>
</action>
```

Multitasking Coordination: Non-Blocking Loops

- Loops and ‘if’ conditionals help with advanced use cases, but…
- Algorithms can block for too long as data sets grow, so…
- XForms has an alternative non-blocking loop pattern:
  - Use ‘if’ instead of ‘while’ to run one iteration of the loop body
  - Use delayed event dispatching to reinvoke *after* yielding

```
<action ev:event="sel-sort-num">
   … <!- Initialize, e.g. make copy of data -->
   <dispatch name="sel-sort-num-background-task" targetid="M"/>
</action>

<action ev:event="sel-sort-num-background-task">
   <action if="instance('SVARS')/items/item">
      … <!-- One iteration of outer loop body -->
      <dispatch name="sel-sort-num-background-task" delay="4"
                     targetid="M“/>
   </action>
   …
</action>
```

Multitasking Coordination: Control Variables

- Attributes can be used to add control variables for multitasking
  - @key – distinguishes which sort algorithm the variables control
  - @state – controls algorithm execution (play, pause, and stop)
  - @priority – controls the amount of delay between iterations

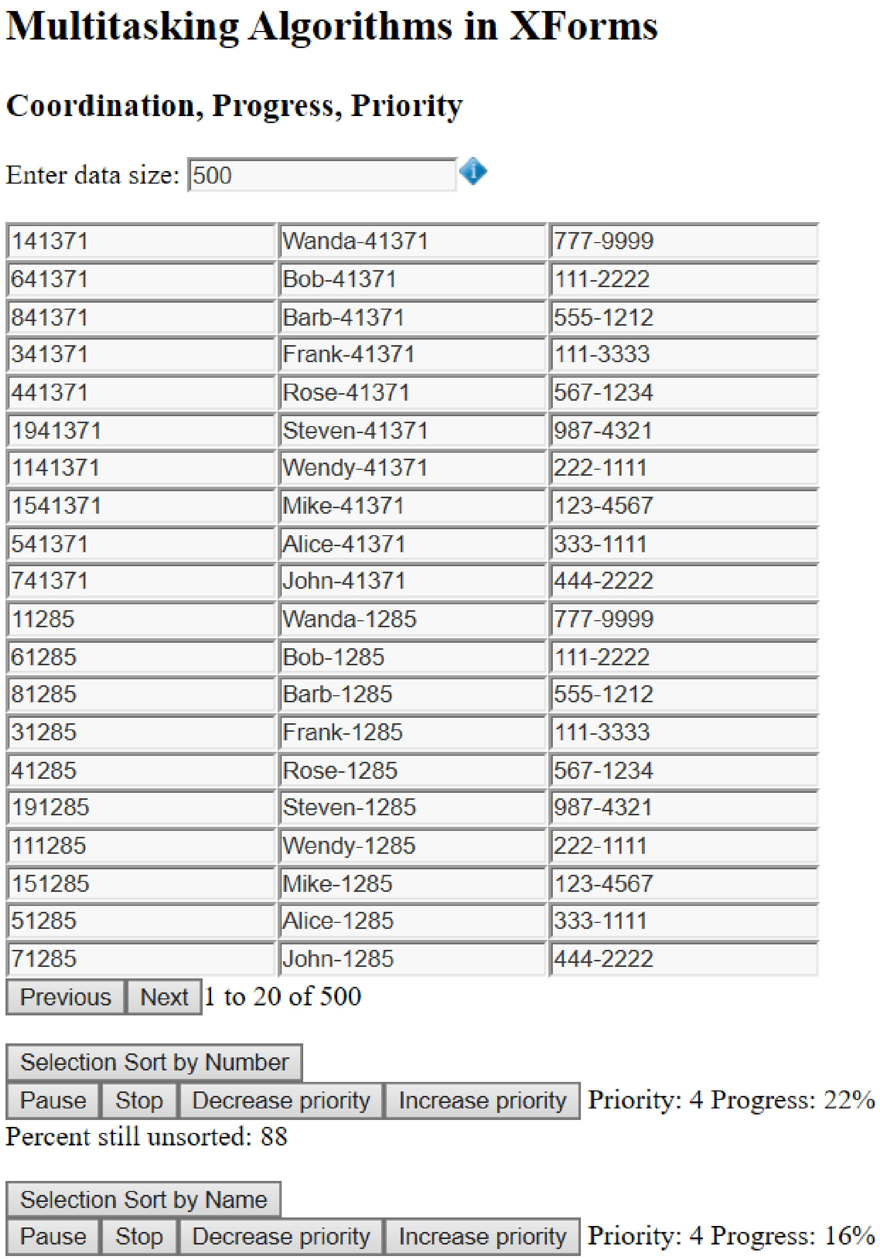
```
<instance id="SVARS">
   <sort xmlns="" key="num | name" state="play | pause | stop" priority="nnn">
      <items/>
      <sorted/>
      <least/>
   </sort>
</instance>
```

Multitasking Coordination: Task List

- To start, insert the algorithm control variables into a task list
- Change the key attribute to indicate the exact sort algorithm
- Adjust XPaths to refer to the copy of the control variables

```
<instance id="TASKS">
    <list xmlns=""/>
</instance>

<action ev:event="sel-sort-num">

    <action if="not(instance('TASKS')/sort[@key='num'])">

        <insert context="instance('TASKS')" origin="instance('SVARS’)"/>
        <setvalue ref="instance('TASKS')/sort[@key='']/@key">num</setvalue>

        <insert context="instance('TASKS')/sort[@key='num']/items"
                   origin="instance('DATA')/items/item"/>

    </action>
    …
</action>
```

Demo of Multitasking and Progress



Multitasking Algorithms in XForms

Coordination, Progress, Priority

Enter data size: 500

141371	Wanda-41371	777-9999
641371	Bob-41371	111-2222
841371	Barb-41371	555-1212
341371	Frank-41371	111-3333
441371	Rose-41371	567-1234
1941371	Steven-41371	987-4321
1141371	Wendy-41371	222-1111
1541371	Mike-41371	123-4567
541371	Alice-41371	333-1111
741371	John-41371	444-2222
11285	Wanda-1285	777-9999
61285	Bob-1285	111-2222
81285	Barb-1285	555-1212
31285	Frank-1285	111-3333
41285	Rose-1285	567-1234
191285	Steven-1285	987-4321
111285	Wendy-1285	222-1111
151285	Mike-1285	123-4567
51285	Alice-1285	333-1111
71285	John-1285	444-2222

1 to 20 of 500

Priority: 4 Progress: 22%

Percent still unsorted: 88

Priority: 4 Progress: 16%

Multitasking Progress, 1

- Add control variables for progress monitoring

```
<instance id="SVARS">
   <sort xmlns="" key="num | name" state="play | pause | stop" priority= "nnn">
      …
      <progress unsortedCount="" unsortedGauss="" totalCount="" totalGauss=""/>
   </sort>
</instance>
```

- Use declarative embellishment to monitor progress… of any and all sort algorithms

```
<bind nodeset="instance('TASKS')/sort/progress/@unsortedCount"
        calculate="count(../../items/item)"/>

<bind nodeset="instance('TASKS')/sort/progress/@totalCount"
        calculate="../@unsortedCount + count(../../sorted/item)"/>
…
```

Multitasking Progress, 2

- Progress is relative to a Gaussian summation of work steps

```
<bind nodeset="instance('TASKS')/sort/progress/@unsortedGauss"
          calculate="(../@unsortedCount div 2) * (../@unsortedCount + 1)"/>

<bind nodeset="instance('TASKS')/sort/progress/@totalGauss"
         calculate="(../@totalCount div 2) * (../@totalCount + 1)"/>

<bind nodeset="instance('TASKS')/sort/progress“
         calculate="concat(round(100 * (1 - @unsortedGauss div @totalGauss)), '%')"/>
```

- Declarative embellishment to pull the progress into the UI

```
<output ref="instance('TASKS')/sort[@key='num']/progress">
    <label> Progress: </label>
</output>
```

Multitasking Priority: Data-Driven Delay

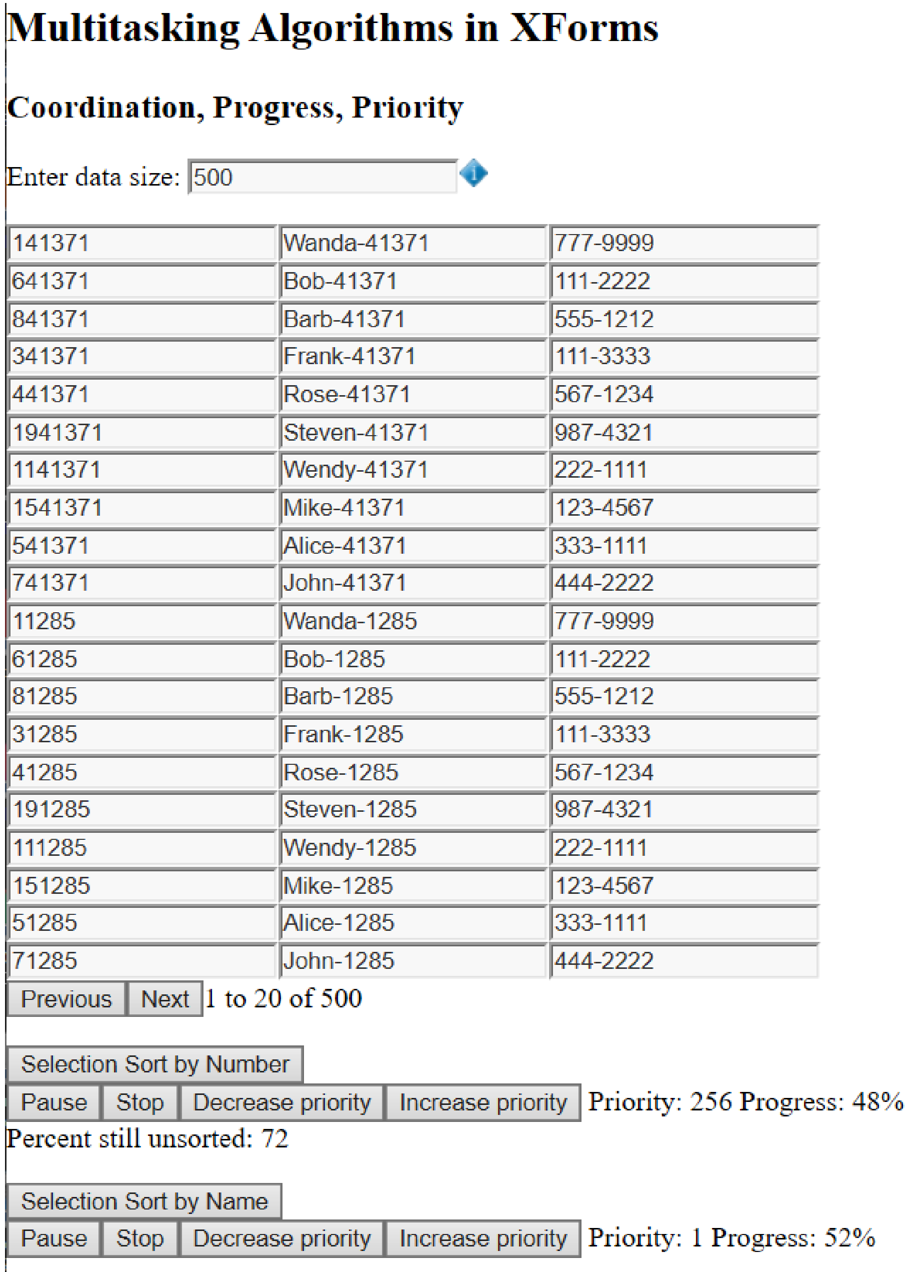
- Delay by declarative infusion in the dispatch action

```
<action ev:event="sel-sort-num-background-task">
    <action if="instance('TASKS')/sort[@key='num']/items/item">
        … <!-- One iteration of outer loop body -->
        <dispatch name="sel-sort-num-background-task" targetid="M">
            <delay value="instance('TASKS')/sort[@key='num']/@priority"/>
        </dispatch>
    </action>
    …
</action>
```

- A longer delay per iteration == lower priority (and vice versa)

```
<trigger>
    <label>Decrease priority</label>
    <setvalue ev:event="DOMActivate"
                  ref="instance('TASKS')/sort[@key='num']/@priority"
                  value=". * 2"/>
</trigger>
```

Demo of Variable Priorities for Tasks



Multitasking Algorithms in XForms

Coordination, Progress, Priority

Enter data size: 500

141371	Wanda-41371	777-9999
641371	Bob-41371	111-2222
841371	Barb-41371	555-1212
341371	Frank-41371	111-3333
441371	Rose-41371	567-1234
1941371	Steven-41371	987-4321
1141371	Wendy-41371	222-1111
1541371	Mike-41371	123-4567
541371	Alice-41371	333-1111
741371	John-41371	444-2222
11285	Wanda-1285	777-9999
61285	Bob-1285	111-2222
81285	Barb-1285	555-1212
31285	Frank-1285	111-3333
41285	Rose-1285	567-1234
191285	Steven-1285	987-4321
111285	Wendy-1285	222-1111
151285	Mike-1285	123-4567
51285	Alice-1285	333-1111
71285	John-1285	444-2222

1 to 20 of 500

Priority: 256 Progress: 48%

Percent still unsorted: 72

Priority: 1 Progress: 52%

Conclusion

- Summary
  - Demonstration of features that make XForms Turing complete and enable handling of advanced data processing use cases
  - Making more powerful imperative script commands with declarative infusion and declarative embellishment
  - A non-blocking loop pattern and task list method that enable multiple tasks to cooperate with each other and the UI thread
  - Methods for task control, progress reporting, and prioritization
- Future Work
  - Upcoming ACM DocEng presentation of recursion and more powerful declarative infusion in a partially declarative quicksort
  - Articulate more XForms use cases for Turing completeness, hybrid imperative/declarative computing, and multitasking
  - Smooth out processing with more sophisticated methods of allocating work to execution time slices