Efficient XML Processing in Browsers

R. Alexander Milowski
ILCC, School of Informatics,
University of Edinburgh
alex@milowski.com

Motivation

- Partially, in response to the anti-XML crowd's complaints about XML in browser applications:
  - XML *slow, inefficient* way to deliver data,
  - JSON is *simpler* and more *directly usable*,
  - and several other *red herrings*.
- **Mostly because I want it!** ...pretty shiny XML objects...
- The reality:

  XMLHttpRequest is insufficient for both XML and JSON delivery.
- Why and what do you do about processing large amounts of XML data **efficiently** in browsers?

Inefficiencies with XMLHttpRequest

Three general deficiencies:

1. If the response is not XML and not characters, there is little support for handling the entity body (e.g. images).
2. If the response is not XML but is characters, treating it as XML or as a sequence of characters may be wasteful.
3. *If the response is XML, the "whole document" intermediary DOM may be wasteful.*

This talk is about concerned with #3.

Strategy

- We want flexibility and choice in our processing model:
  - whole document, subsetting, multiple DOMs,
  - view porting, filtering,
  - or just a stream of events.
- We'll replace XMLHttpRequest and:
  - Keep the request formulation,
  - Remove the "whole document" treatment of the response,
  - Add event-oriented processing of the XML.

The XMLReader Interface

- Shares a lot in common with XMLHttpRequest for making the request:
  - send, open, overrideMimeType, setRequestHeader, etc.
  - request model is the same,
- added a parse(in DOMString xml) method for completeness,
- added an onxml event listener attribute for receiving XML,
- added an "xml" event type for addEventListener()

XML Events

- Events for:
  - start/end document,
  - start/end element,
  - characters,
  - processing instructions,
  - comments
- Events are flattened - one interface for all of them.
- Provides:
  - full names, attributes,
  - namespace declarations, base uri, values, ...

Event Interfaces

```
interface XMLItemEvent : Event {
   readonly attribute unsigned short itemType;
   readonly attribute DOMString prefix;
   readonly attribute DOMString localName;
   readonly attribute DOMString namespaceURI;
   readonly attribute DOMString uri;
   readonly attribute DOMString value;
   DOMString getAttributeValue(...);
   readonly attribute Array namespaceDeclarations;
   readonly attribute Array attributeNames;
}

interface XMLName {
   readonly attribute DOMString prefix;
   readonly attribute DOMString localName;
   readonly attribute DOMString namespaceURI;
}
```

Simple Usage Example

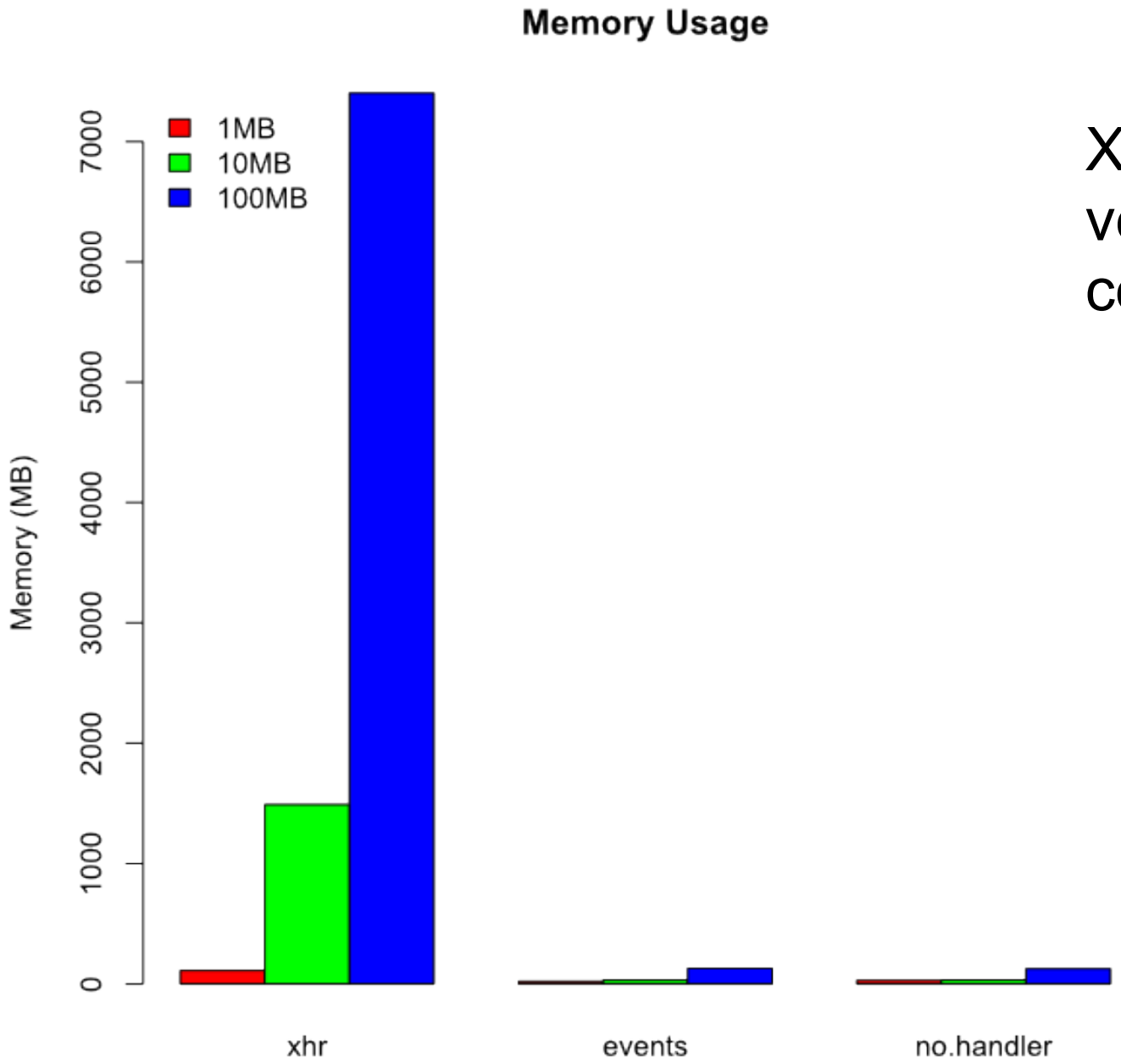
```
var reader = new XMLReader();
var links = [];

reader.onxml = function(e) {
   if (e.itemType==XMLItemEvent.START_ELEMENT &&
       e.localName=="a") {
      var href = e.getAttributeValue("href");
      if (href) {
         links.push(href);
      }
   }
}
reader.open("GET","http://www.milowski.com/");
reader.send();
```

WebKit Implementation

- **...and so I built it!**
- Hacked up XMLHttpRequest,
- in WebKit,
- and swapped out WebKit in Safari.
- (I am using it now)

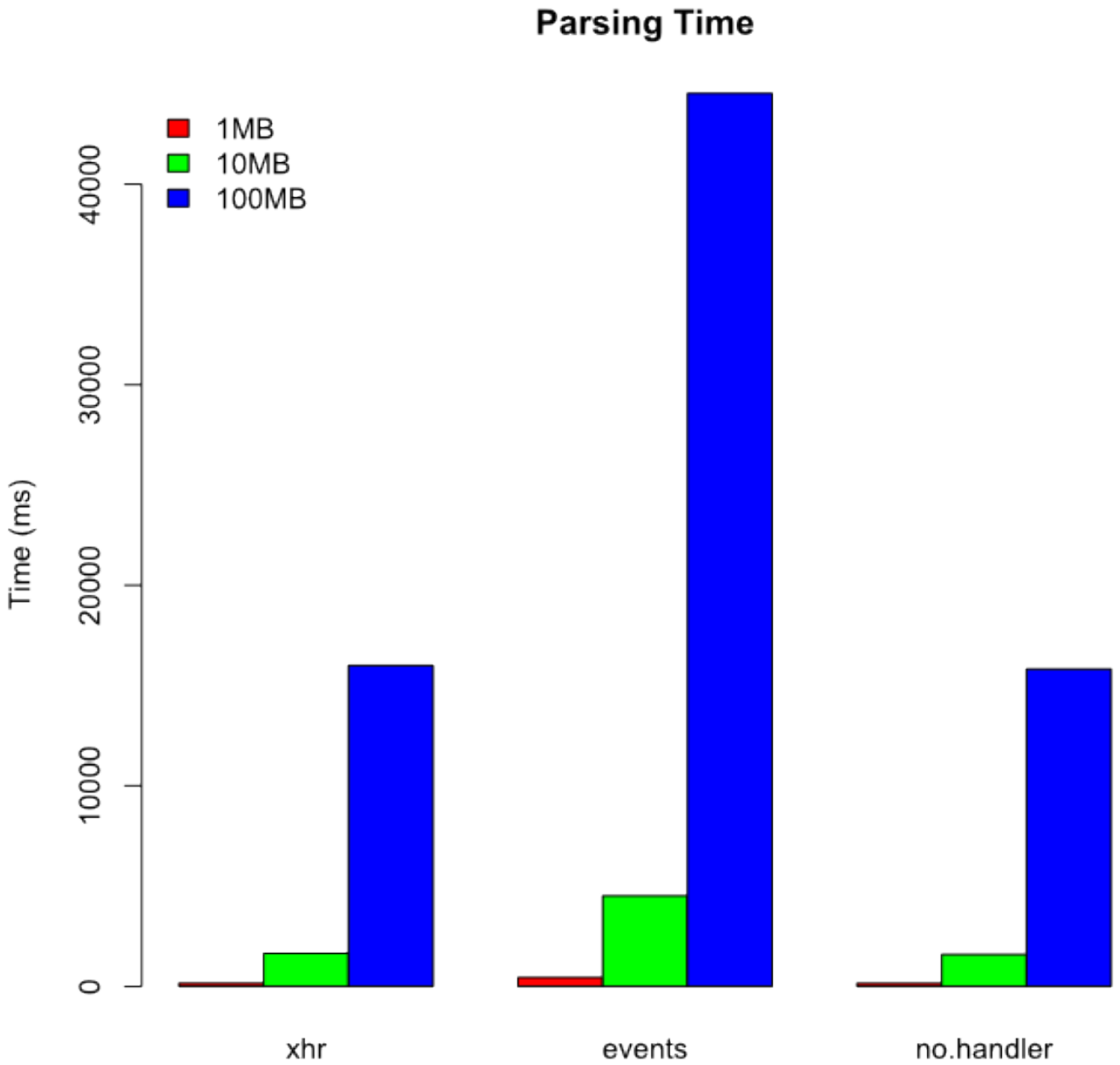
Performance - Memory



XMLReader performs very well - nearly constant memory!

Note: The largest xhr test consumed all memory on my system before crashing.

Performance - Parsing Time



- 2.75 times penalty for delivering events,
- reduced to 2 times if events are chunked,
- overall application time may be faster.

Have your XML and eat your JSON too!

- Event-oriented JSON binding:
  - avoids building intermediary DOM,
  - allows filtering, view porting, subsetting,
  - typing can be context driven,
- Simple example developed in JavaScript (182 lines),
- Efficiency can be increased by "going native".
- The "Live" version is here.

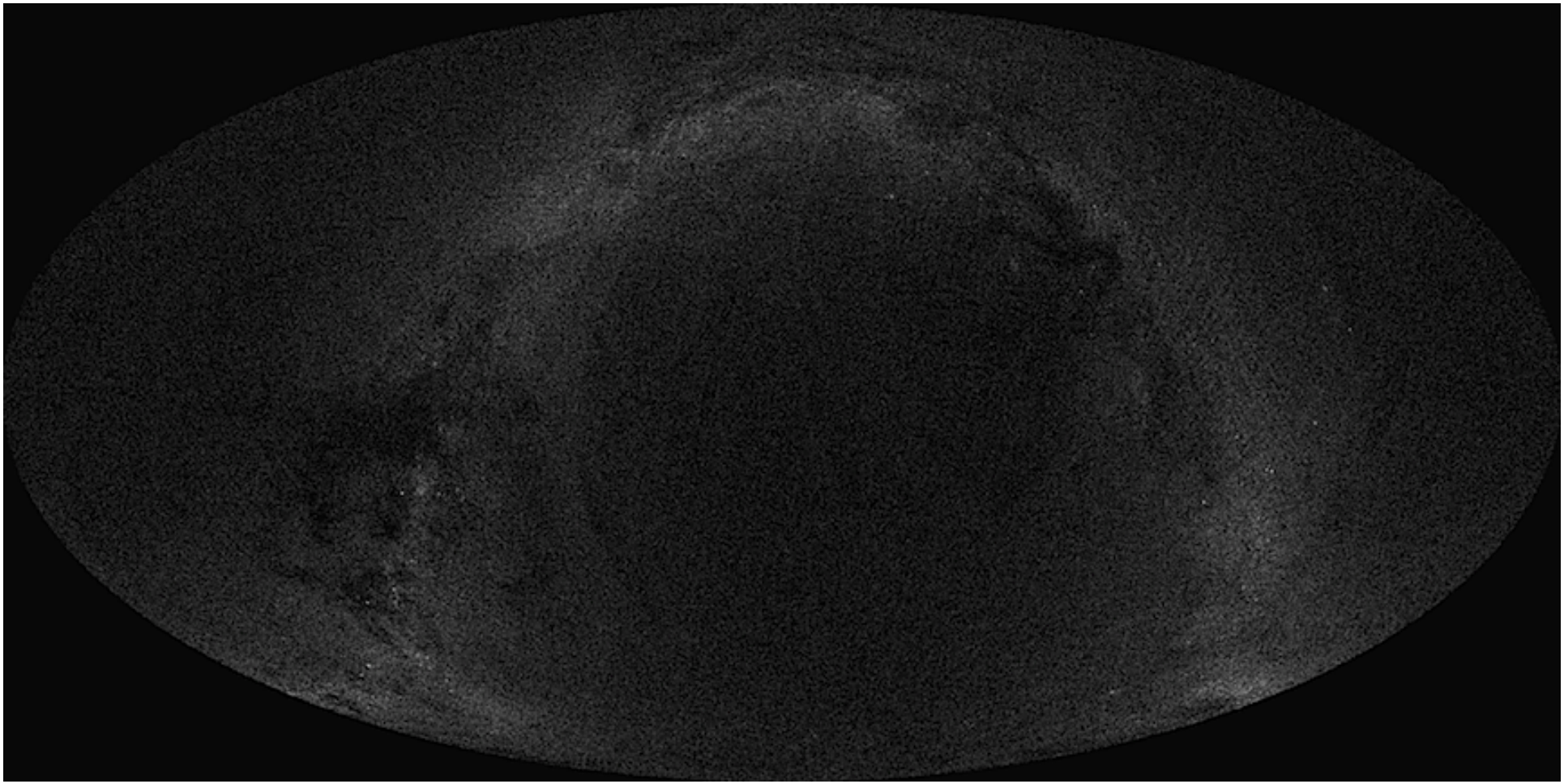
```
<order count="1">
<item>
<price>12.
50</price><quantity>2</quantity>
<description>...</description>
</item>
</order>
```

```
{ "count" : 1,
  "item" : {
    "price" : 12.5,
    "quantity": 2,
    "description": [ ... ]
  }
}
```

Note: Must have "enhanced browser"

Mollweide Projection of the ACT Star Catalog

A synthetic image the "visible" stars in the celestial sphere



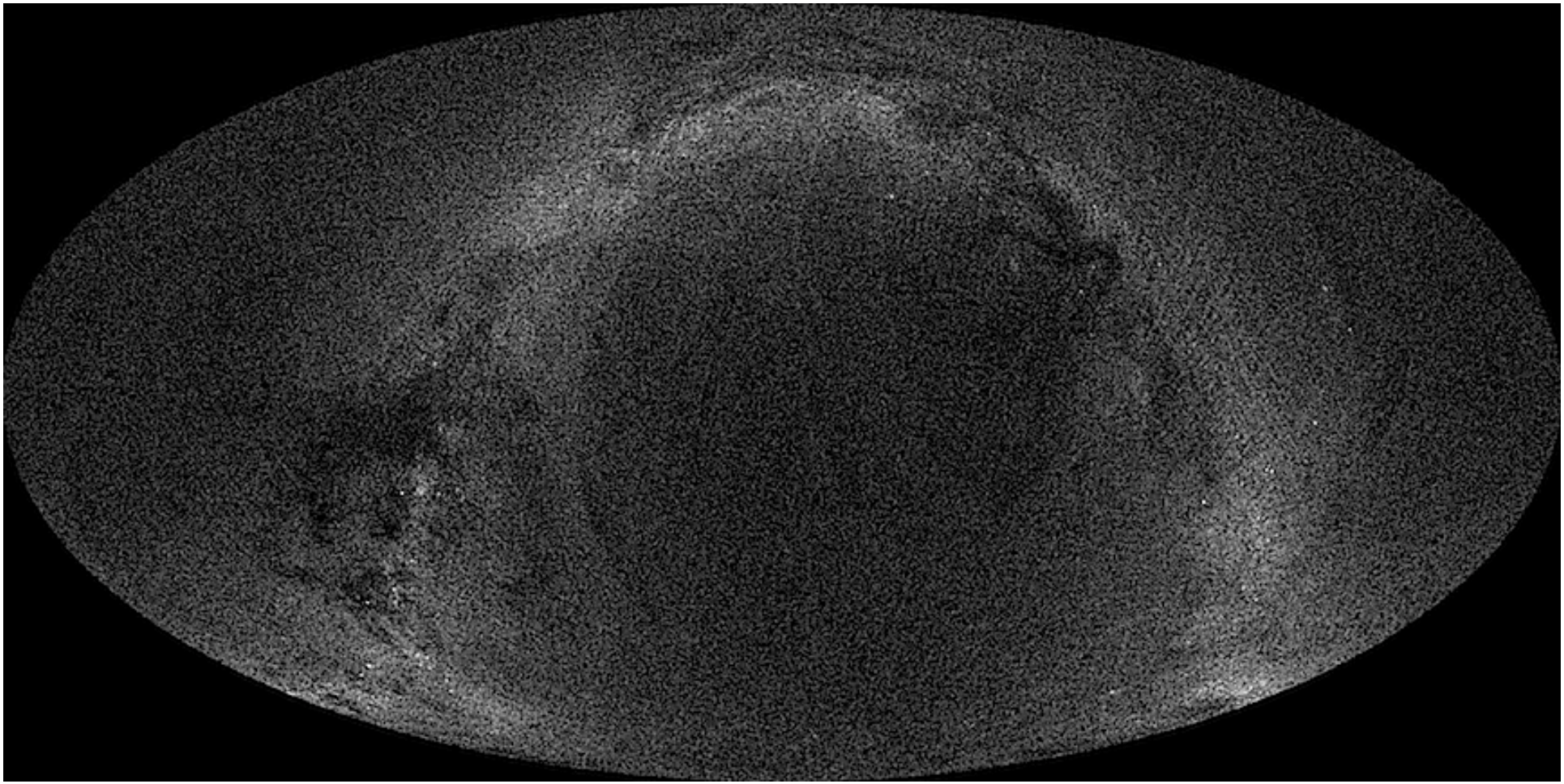
USNO Cone Service + HTML5 canvas + XMLReader + 1h20m

XMLReader is slightly faster than XHR

~1 GB of XML

Mollweide Projection of the ACT Star Catalog

A synthetic image the "visible" stars in the celestial sphere



USNO Cone Service + HTML5 canvas + XMLReader + 1h20m

XMLReader is slightly faster than XHR

~1 GB of XML

Mollweide Projection of the ACT Star Catalog

A synthetic image the "visible" stars in the celestial sphere

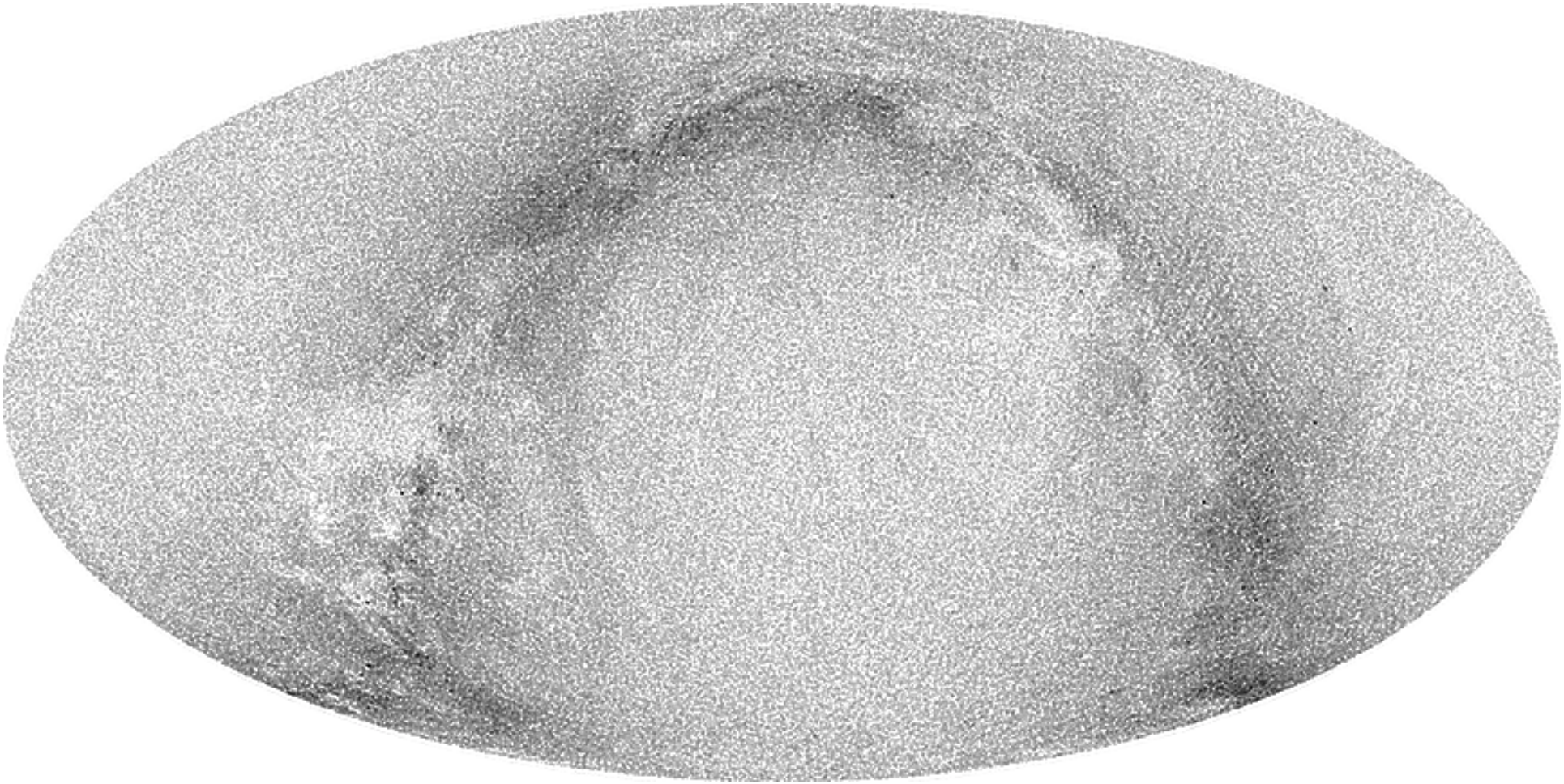
USNO Cone Service + HTML5 canvas + XMLReader + 1h20m

XMLReader is slightly faster than XHR

~1 GB of XML

Mollweide Projection of the ACT Star Catalog

A synthetic image the "visible" stars in the celestial sphere



USNO Cone Service + HTML5 canvas + XMLReader + 1h20m

XMLReader is slightly faster than XHR

~1 GB of XML

What next?

- More information, the paper, slides, sample code, etc. is available from:

  http://www.milowski.com/research/xmlreader/
- WebKit bug (patch) #57145.
- For the courageous, you can build WebKit yourself.
- Other ideas:
  - Firefox extension?
  - universal plugin for all browsers?
- It's open source so ...