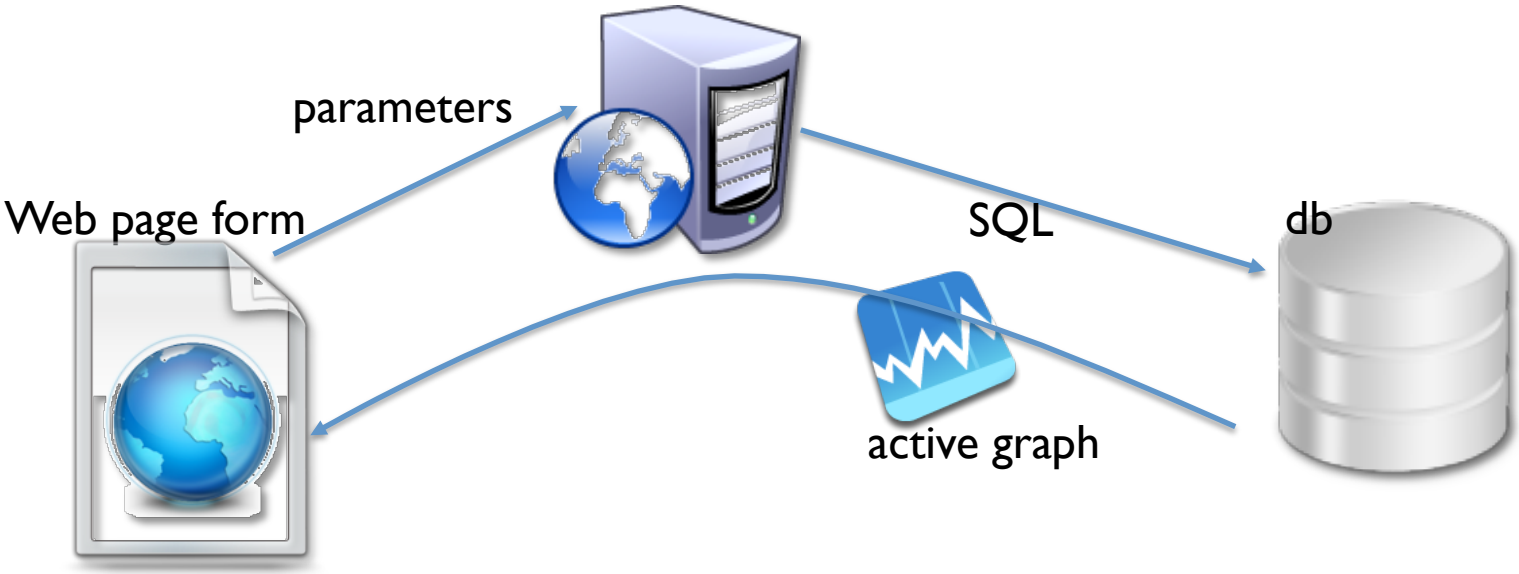
Ajax, XSLT and SVG
Displaying ATLAS conditions data with new web technologies

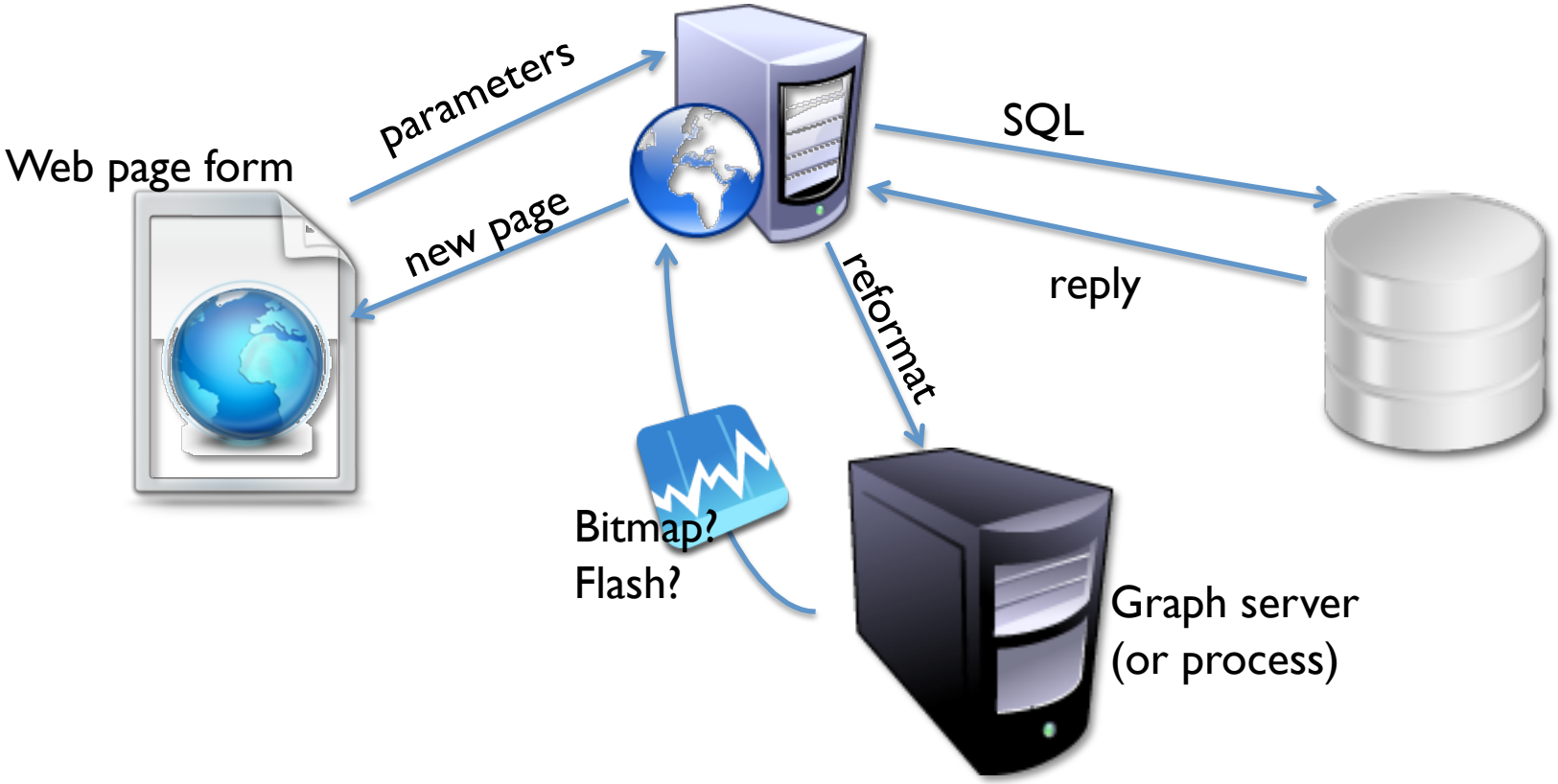
Shaun Roe

Aim

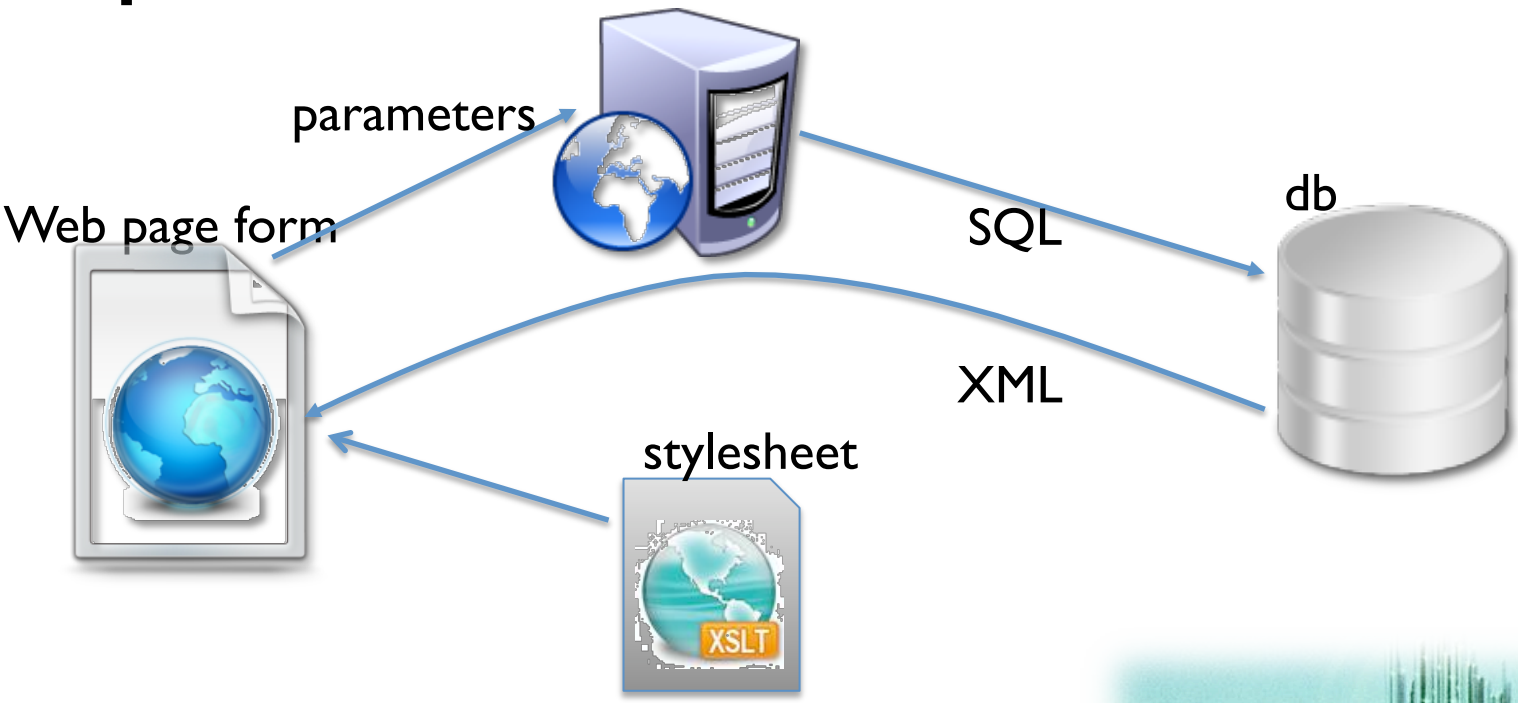


Don't want the web page to 'blink' => insert the graph to the page in memory
Want the graph to have hyperlinks, where appropriate=> ?
Would like the output format to be flexible => separate presentation from information

Yesterday's alternatives



A plan



So I need:

1. A way of getting xml from the database
2. A stylesheet to format xml so it looks like a graph
3. A way of injecting that graph…

XML from Oracle

(I'm dealing with Oracle. For MySQL, see xaware.org.)

SELECT DBMS_XMLGEN.GETXML('query') XML FROM DUAL

It's (almost) that simple. In practice, have to be careful about quotes:

```
$query = "SELECT TO_CHAR(A.TS, ''YYYY-MM-DD HH24:MI:SS'') AS …";
```

The reply

```
<ROWSET>
 <ROW>
  <ELEMENT_ID>1065554581272</
ELEMENT_ID>
  <COMMENT_>SCT BAR PS Q2 B303
ch8000 lay0 phi12 eta06 HVchVolt</
COMMENT_>
  <VALUE_NUMBER>5.0</VALUE_NUMBER>
  <DATETIME>11-AUG-08
11.27.41.931000000 AM</DATETIME>
  <TIMESTAMP>1218454061.931</
TIMESTAMP>
 </ROW>
 <ROW>
  <ELEMENT_ID>106555458127
```

Great! Is there more like this?

- Form sub nodes with subqueries using oracle 'cursor' function
- Stored functions allow one to page the xml
- Stylesheets can be stored and applied on the DB.
- XMLDB package allows one to create an XML server *on the DB* which responds to http calls.

Styling and Transforming XML

CSS Cascading Style Sheets

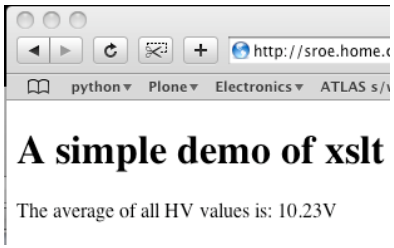
```
ROWSET {
  font-family:arial, helvetica,
sans-serif;
  text-align:left;
  margin:10px;
  display: table;
  border-width: 1px;
  border-style: solid;
 }
 ROW {
  display:table-row;
  visibility: visible;
  font-size: 10pt;
 }
 ROW > * {
  display:table-cell;
  border-width: 1px;
  border-style: solid;
  padding: 4px;
 }
```

Element ID	Value	Date and time	Seconds of Epoch
1065554581272	5.0	11-AUG-08 11.27.41.931000000 AM	1218454061.931
1065554581272	5.0	11-AUG-08 05.47.03.676000000 PM	1218476823.676
1065554581272	20.0	11-AUG-08 05.56.50.055000000 PM	1218477410.055
1065554581272	5.1	11-AUG-08 05.58.50.100000000 PM	1218477530.1
1065554581272	20.0	11-AUG-08 06.04.20.124000000 PM	1218477860.124
1065554581272	19.9	11-AUG-08 06.35.50.010000000 PM	1218479750.01
1065554581272	12.9	11-AUG-08 07.02.38.120000000 PM	1218481358.12
1065554581272	5.0	11-AUG-08 07.02.48.120000000 PM	1218481368.12

XSLT XML Stylesheet Language for Transformations

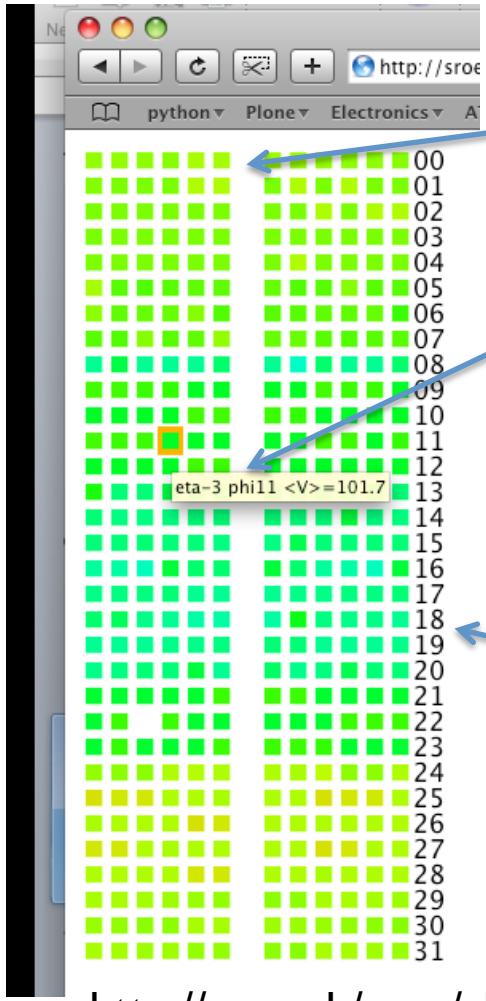
Allows radical restructuring; the stylesheet is a complete programming language.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/
Transform" version="1.0">
<xsl:output method="html" indent="yes"/>
<xsl:template match="ROWSET">
 <html><head><title>Simple XSLT</title></
head><body><h1>A simple demo of xslt</h1>
  <xsl:variable name="sumValues" select="sum(ROW/
VALUE_NUMBER)"/>
  <p>The average of all HV values is:
   <xsl:value-of select="format-number($sumValues div
count(ROW),'###.##')"/>V</p>
 </body></html>
</xsl:template>
</xsl:stylesheet>
```



http://cern.ch/sroe/chep/HVcss.xml
http://cern.ch/sroe/chep/HVxsl0.xml

Example of XSLT applied

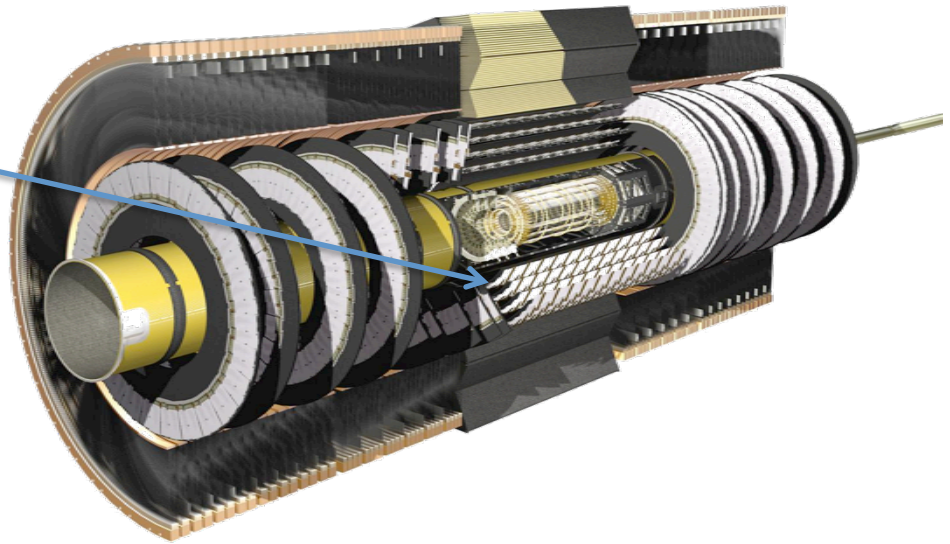


Blocks are printed; their colour and position are determined by a CSS.

For each element, the time average has been calculated and inserted as a tooltip

44 LOC, excluding CSS
Everything calculated by the browser

Bias voltage



http://cern.ch/sroe/chep/HVxsl2.xml

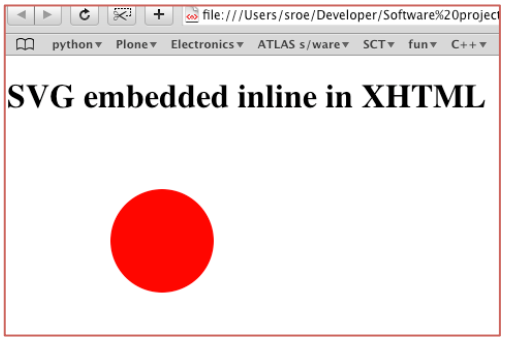
SVG: A Graphical web standard

Wouldn't it be great if this worked:

```
<html>
  <head>
    <title>This is a circle</title>
  </head>
  <body>
    <h1>Show a circle</h1>
    <graph width="300" height="200">
      <circle cx="150" cy="100" r="50"
fill="red"/>
    </graph>
  </body>
</html>
```

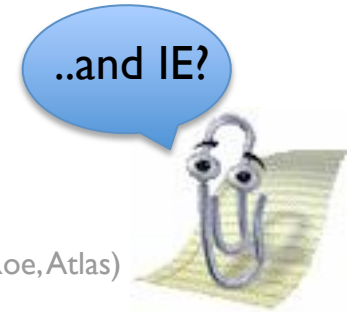
This does work:

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:svg="http://www.w3.org/2000/svg">
  <head>
    <title>SVG embedded inline in XHTML</title>
  </head>
  <body>
    <h1>SVG embedded inline in XHTML</h1>
<svg:svg width="300" height="200">
<svg:circle cx="150" cy="100" r="50" fill="red/>
    </svg:svg>
  </body>
</html>
```



Scalable Vector Graphics

- A W3C recommendation
- Understood natively by Firefox, Safari, Opera
- Probably on your mobile phone
- Almost certainly in your digital TV



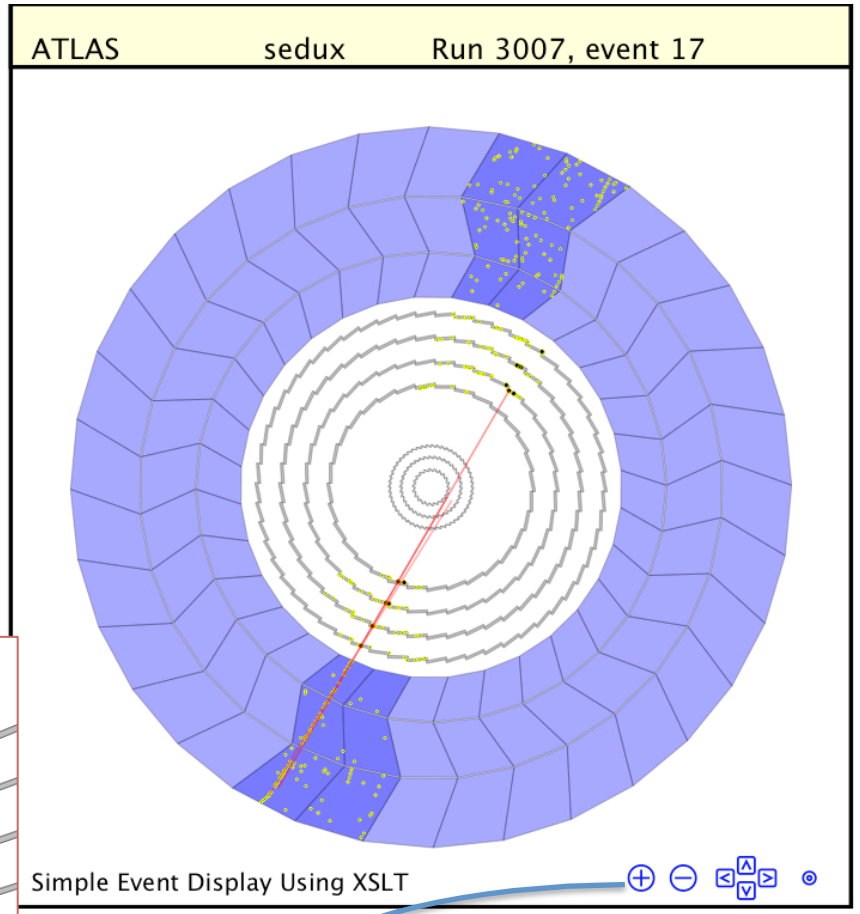
(Vector Markup Language)

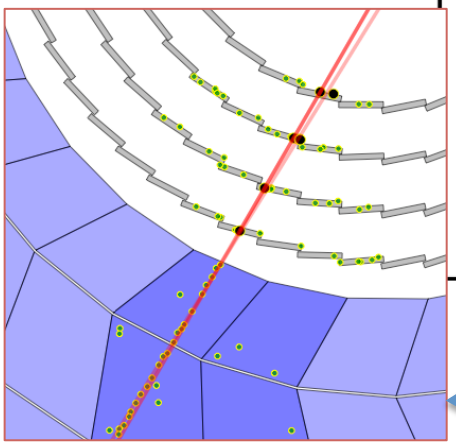
Extreme SVG

http://cern.ch/sroe/display/example.xml

JiveXML

```
<?xml version="1.0"?><?xml-stylesheet
type="text/xsl" href="JiveXML_event.xsl"?
><?ATLAS Release: "None"?><!DOCTYPE Event
SYSTEM "event.dtd"><Event version="None"
runNumber="3007" eventNumber="17"
dateTime="2006-06-20 11:01:54 CEST"><STC
count="138" storeGateKey=""><barcode>0 0 0
0 0 0 0 0 0 0 00 0 0 0 0 0 0 0 0 0 00 0 0 0
0 0 0 0 0 0 0 00 0 0 0 0 0 0 0 0 0 00 0 0 0 0
0 0 0 0 0 0 00 0 0 0 0 0 0 0 0 0 00 0 0 0 0 0
0 0 0 0 0 00 0 0 0 0 0 0 0 0 0 00 0 0 0 0 0 0
0 0 0 0 00 0 0 0 0 0 0 0 0 0 00 0 0 0 0 0 0 0
0 0 00 0 0 0 0 0 0 0 0 0 00 0 0 0 0 0 </
barcode><etaModule>-5 5 -5 -4 -3 1 3 3 3 -2
-1-1 3 3 3 3 -3 4 -3 -1 -1 1…
```



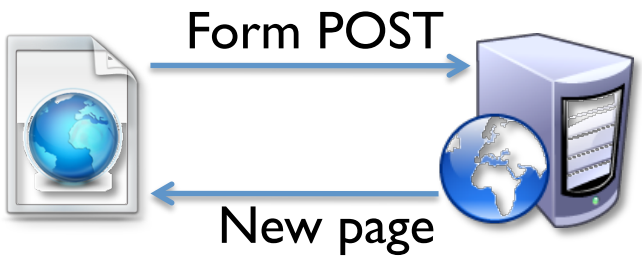


zoom

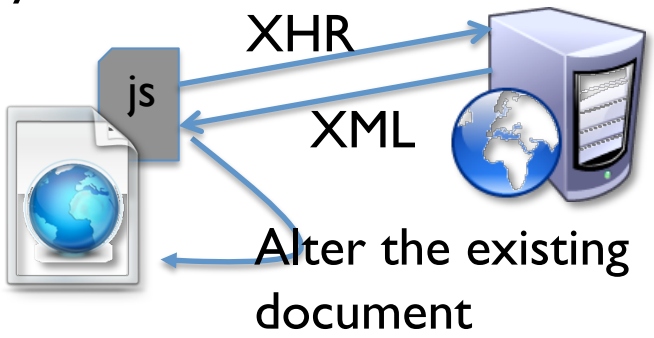
Ajax

Asynchronous Javascript And XML

A way of changing web pages in memory



Xml Http Request



Forming an XHR is browser specific, so use a common javascript library: prototype

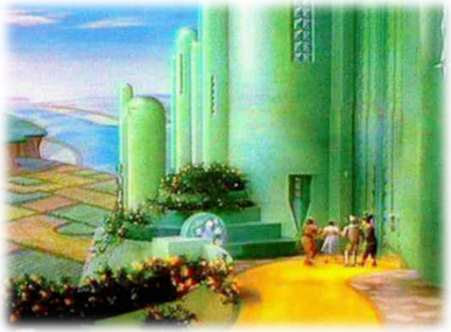
```
 <form name="formInputs" id="formInputs"
action="./php/tim.php" method="get"><input
type="hidden" name="method" id="method"
value="get_times"/>
.
.
.
<div id="results">
<!-- insert results here -->
</div>
```

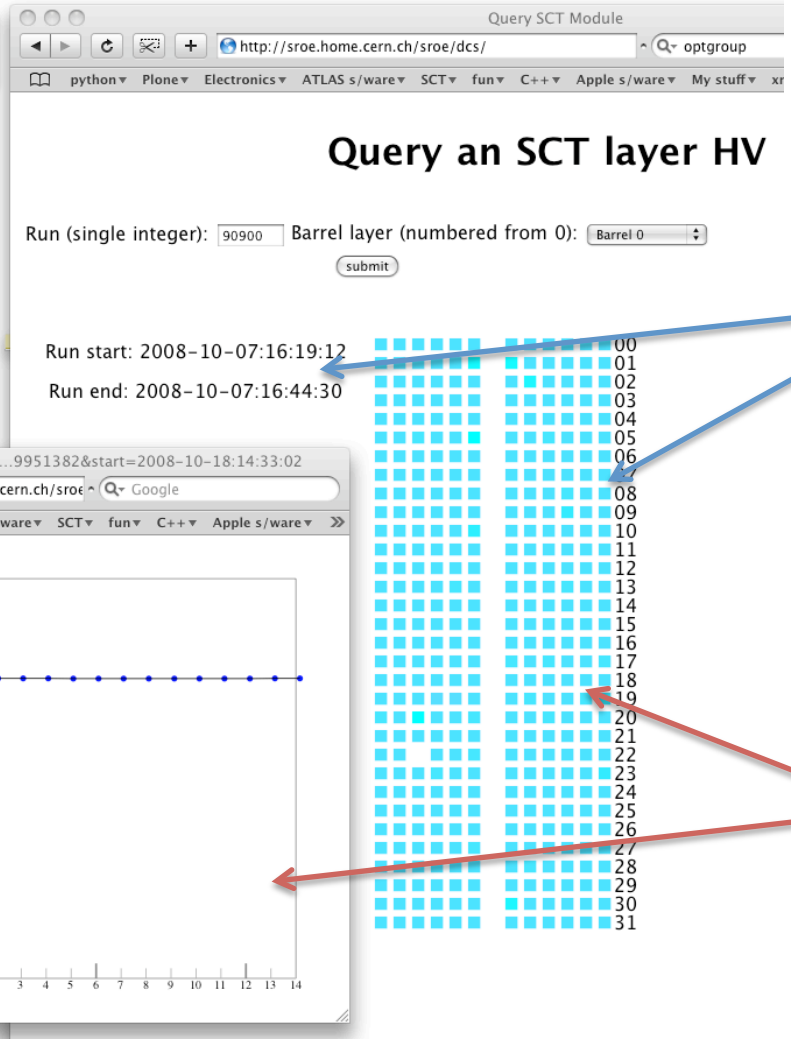
```
  function bindForm(){
    $('formInputs').observe('submit',
function(e){
      e.stop();
      new Ajax.Request( this.action, {
       method: 'get',
       parameters: this.serialize() ,
       onSuccess: function(r){…

var iframe=new Element('iframe',{'src': url,
'height':'800px', 'frameborder':'0'});
      $('results').update(iframe);
…}
```

http://blog.jjg.net/

http://www.prototypejs.org/

Putting it together





Inserted via Ajax

XML, styled and transformed

Variations

I've shown simply providing an XML which is associated with an XSLT stylesheet; the browser does the transformation. XSLT is now common enough that the transform can be done at various stages:

In Oracle: one can store stylesheets in the database and request one of them to be used before sending the data.

On the server: PHP, Python, Perl all have interfaces to the XSLT engine.

Explicitly from javascript: An Ajax request can be made to retrieve a stylesheet to be applied.

These alternatives allow one to associate different stylesheets 'on the fly' and parametrize them (providing arguments to them).

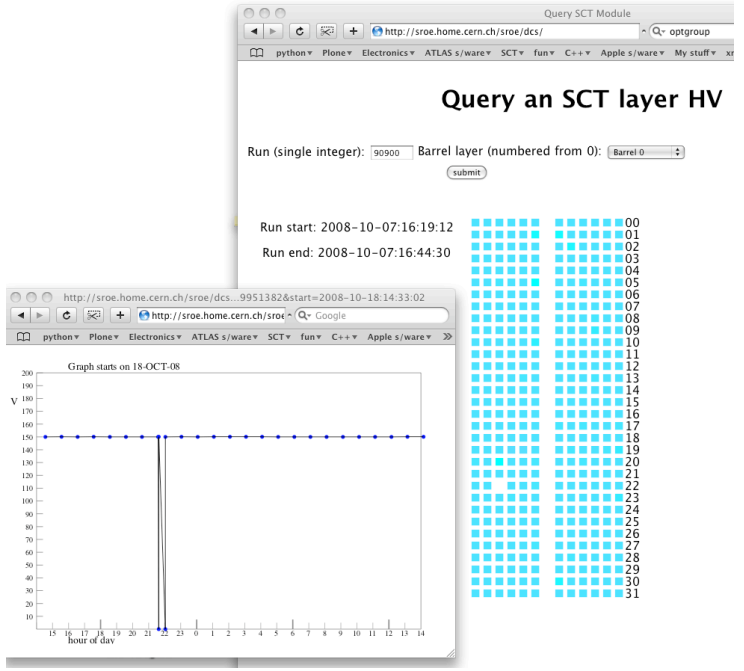
Conclusion

The combination of XML, XSLT and SVG provides a straightforward way to produce 'active' graphics, naturally separating information from presentation

Ajax provides a seamless interaction between web page and server

The whole is a code-efficient way to produce graphics from database queries

Lines Of Code (excluding css):
HTML: 74
Js: 46
PHP: 108 (of which SQL is ~20 lines)
XSL: 44 (table) + 133 (graph) = 177
Total: 405 LOC



(Thanks)