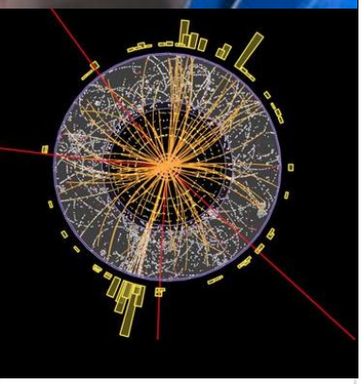
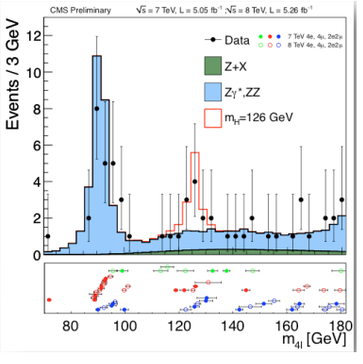
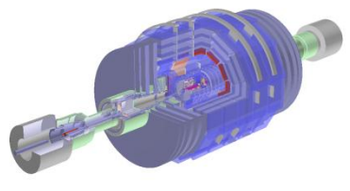






ROOT I/O in JavaScript

Reading ROOT files from any web browser

ROOT Users Workshop 2013

Bertrand Bellenot
root.cern.ch

Introduction

- How to share thousands of histograms on the web, without having to generate picture files (gif, jpg, …)?
- How to easily share a ROOT file?
- How to browse & display the content of a ROOT file from any platform (even from a smartphone or tablet)?
- And obviously, all that without having to install ROOT anywhere?

Requirements

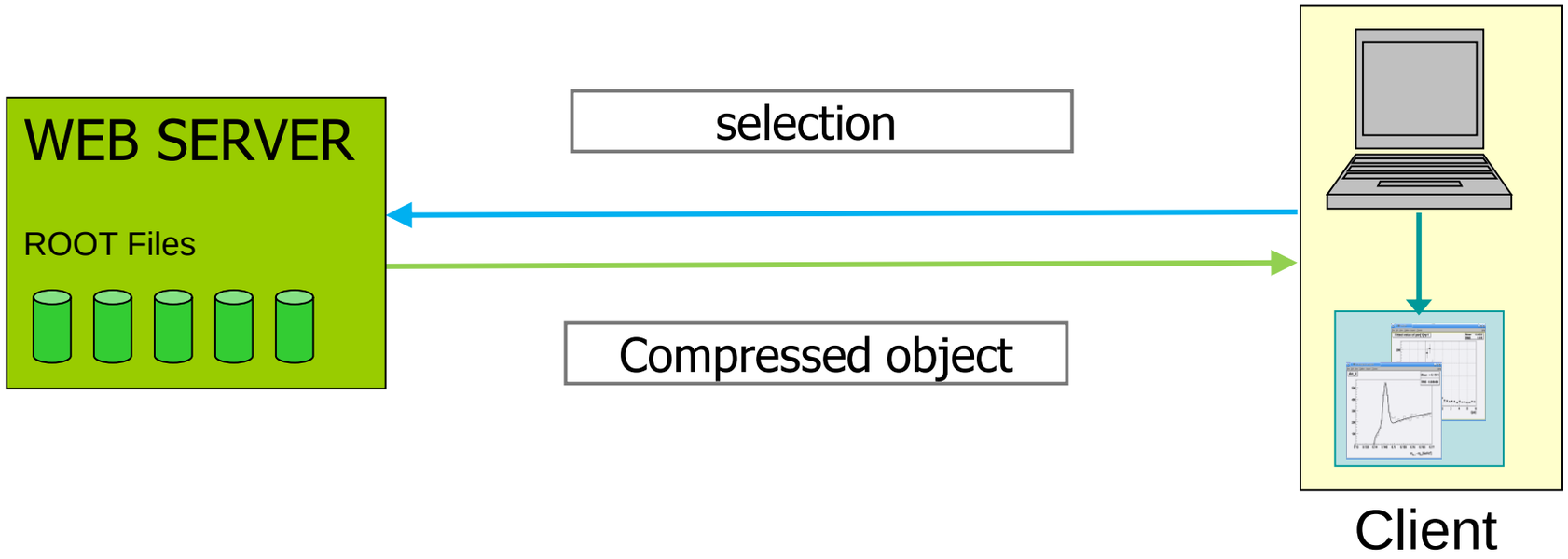
- The solution should be:
  - Portable: use available web browser
  - Lightweight: no library or application to install
  - Easy to use (user side)
  - Easy to extend and maintain (developer side)
  - Fast, with a small memory footprint

Targeted scope

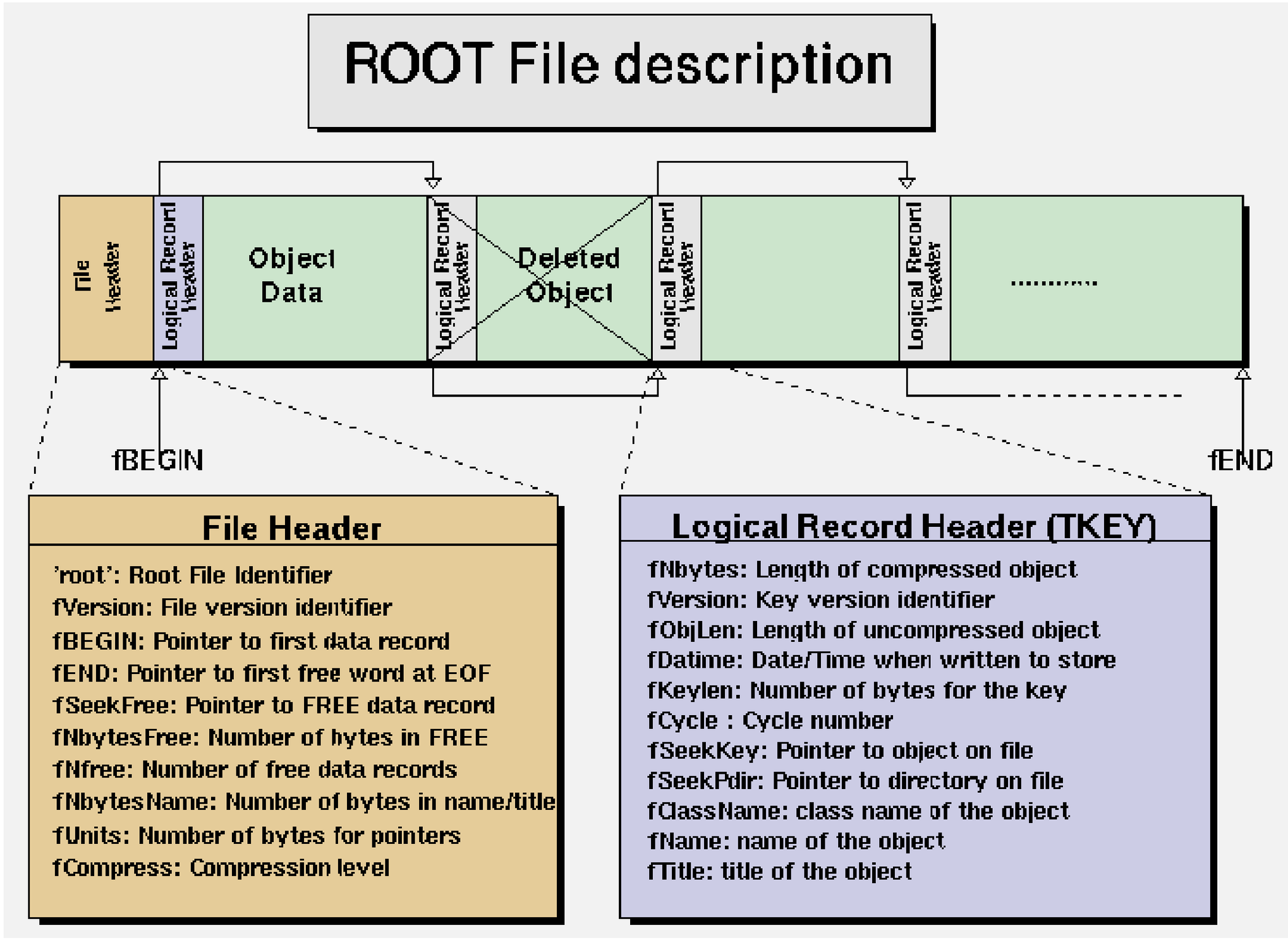
- Not a replacement of ROOT:
  - No tree analysis
  - No fitting
  - No editing
- Visualization only:
  - Online monitoring
  - Publication sites (Elsevier is going to use it)
  - …

Solution

- HTML & JavaScript: JSROOTIO.js
  - Copy the ROOT file on any plain web server
  - Data is transferred over the web
  - Visualization happens on the client side



ROOT I/O


ROOT File description
File Header
Logical Record Header
Object Data
Logical Record Header
Deleted Object
Logical Record Header
Logical Record Header
.....
fBEGIN
fEND
File Header
'root': Root File Identifier
fVersion: File version identifier
fBEGIN: Pointer to first data record
fEND: Pointer to first free word at EOF
fSeekFree: Pointer to FREE data record
fNbytesFree: Number of bytes in FREE
fNfree: Number of free data records
fNbytesName: Number of bytes in name/title
fUnits: Number of bytes for pointers
fCompress: Compression level
Logical Record Header (TKEY)
fNbytes: Length of compressed object
fVersion: Key version identifier
fObjLen: Length of uncompressed object
fDatime: Date/Time when written to store
fKeylen: Number of bytes for the key
fCycle : Cycle number
fSeekKey: Pointer to object on file
fSeekPdir: Pointer to directory on file
fClassName: class name of the object
fName: name of the object
fTitle: title of the object

Reminder: the streamer info

- A TStreamerInfo object describes a persistent version of a class.
- A ROOT file contains the list of TStreamerInfo objects for all the class versions written to this file.
- A TStreamerInfo is a list of TStreamerElement objects (one per data member or base class)
- A TStreamerElement describe a data member or a base class (e.g. type, name)

Reading the file

- When opening the file:
  - Read the list of streamer info
  - Read the list of keys and display them in a list tree
- When the user select an item in the list tree (and only then)
  - Read the compressed buffer from the file
  - Inflate the buffer
  - Decipher the object from the inflated buffer using its streamer info

Reading the file

- Use the XMLHttpRequest AJAX API to perform the HTTP HEAD and GET requests
- This API is highly browser dependent:
  - On IE, the binary data is in its responseBody data member (VBScript format), and has to be converted into a JavaScript string
  - On other browsers, the data can be in response, mozResponse, mozResponseArrayBuffer, or responseText...
  - Thanks to Ioannis Charalampidis, who kindly provided a working cross-browser solution

Downloading & handling binary data

- Using HTTP byte range (available in HTTP/1.1) to download only a single compressed object when the user wants to read it
- Minimizes data transfer and memory usage
- Compressed (zipped) objects are in binary format

- JavaScript has very little support for raw binary data
- Binary data is simply stored in a JavaScript string
- Accessing a single byte is easy:

```
byte = string.charCodeAt(index);
```

Reading the keys

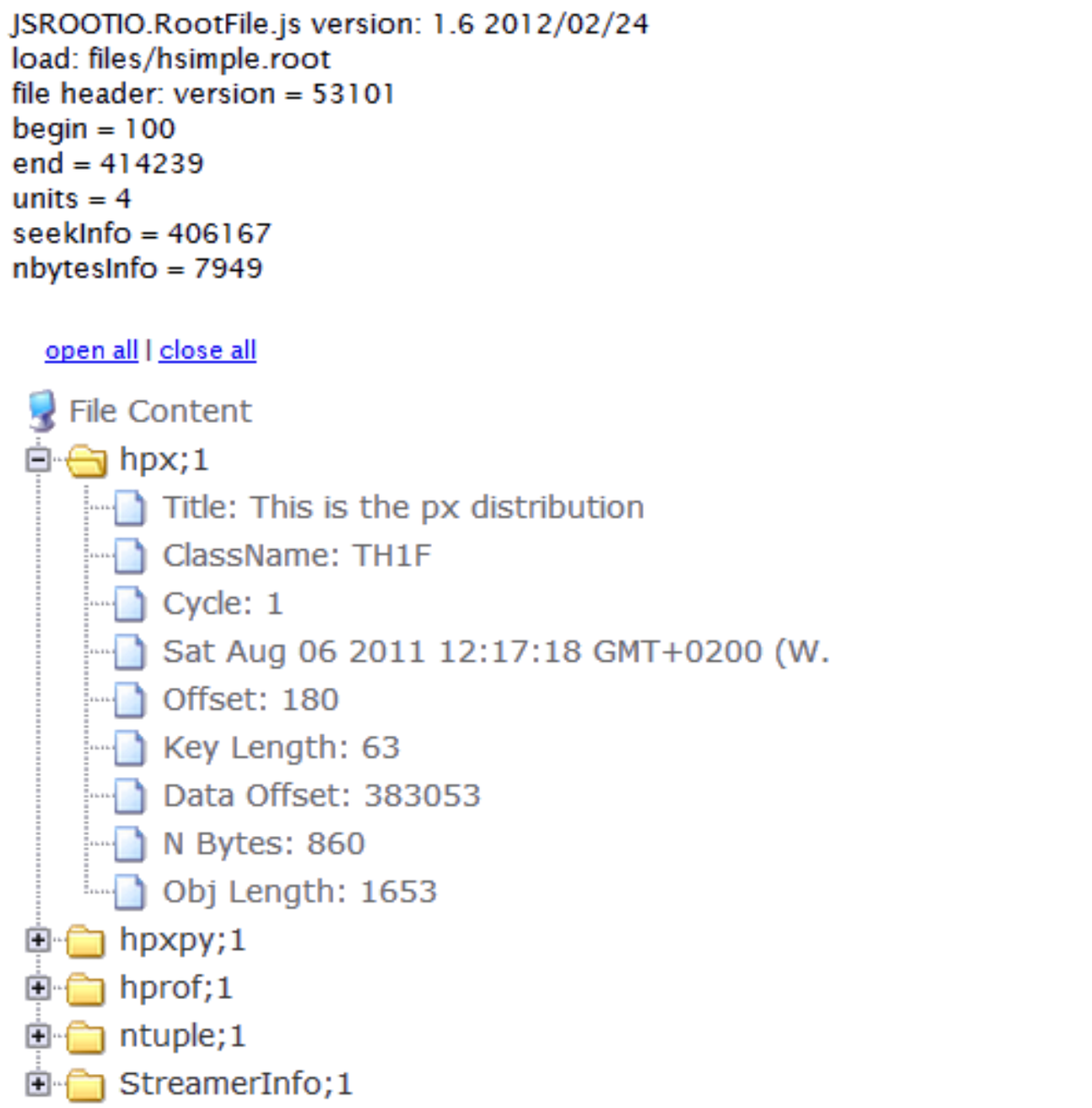
- The keys are not compressed
- They contain basic information on the object they describe, like its name and its type
- First step was quite easy, starting from already working code written by Axel Naumann
- Formatting and displaying the keys is done with a JavaScript tree menu

File header and key information

Screenshot of the file header and the list of keys contained in hsimple.root

The hpx key is open, showing the information describing the TH1F object in the file

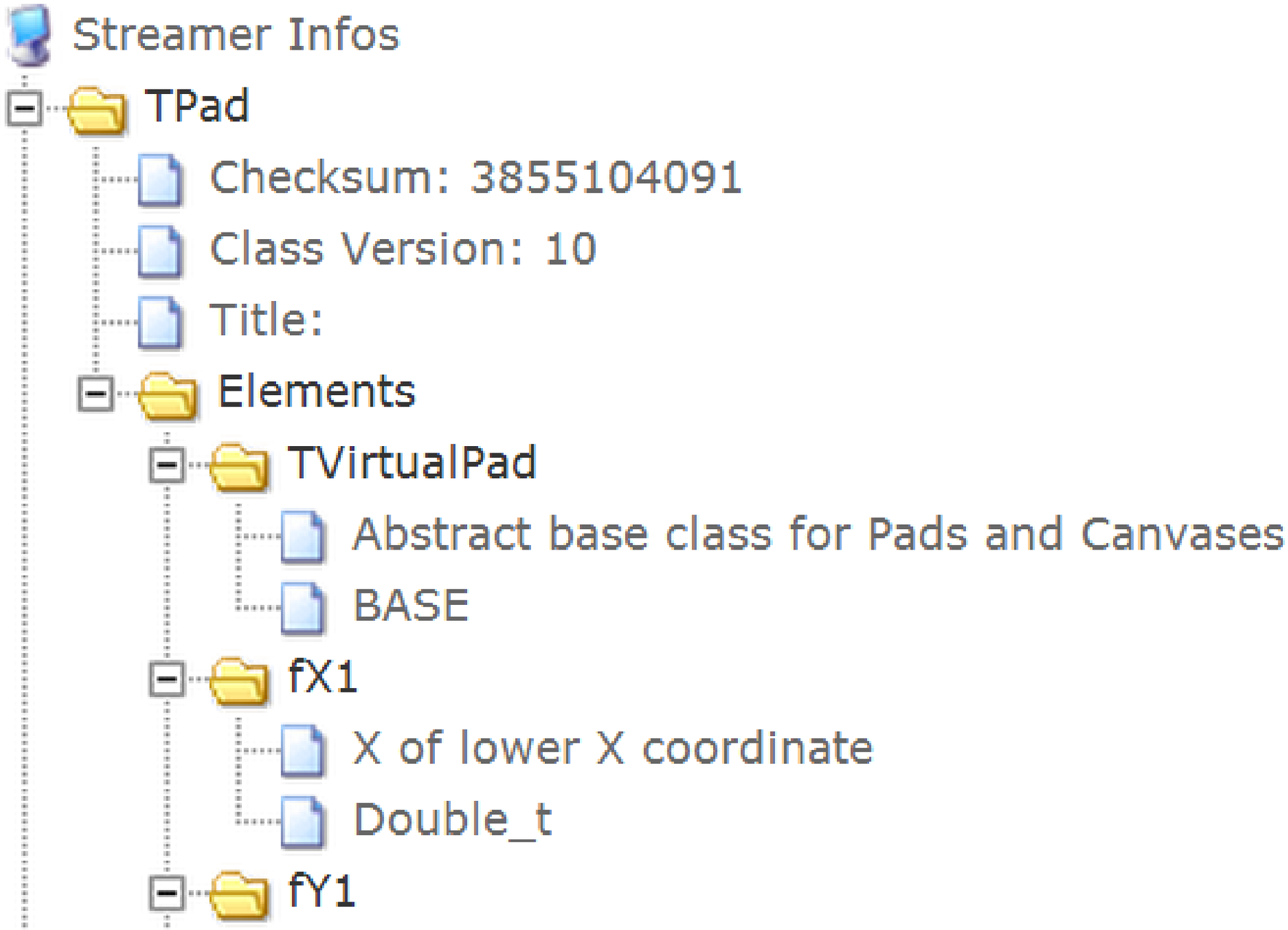
(displayed for debugging purpose only)



Inflating & decoding the streamer info

- Inflating (unzipping) the buffers required:
  - A JavaScript version of zlib's inflate function from: http://www.onicos.com/staff/iz/amuse/javascript/expert/inflate.txt
- Implementing the streamer info functionality in JavaScript involved:
  - reverse engineering and parallel debugging of C++ and JavaScript
  - valuable help from Philippe Canal
- Streamer info can be displayed for educational / informational purposes

Streamer info visualisation



Reading objects, first version

- At the beginning, the classes' streamers were hard-coded. This approach has several issues:
  - Streamers must be updated with every change in the original class
  - Add a new streamer for every new class
  - The library is growing with every new streamer
- The only (partially) supported classes were TH1, TH2, TGraph, and TProfile

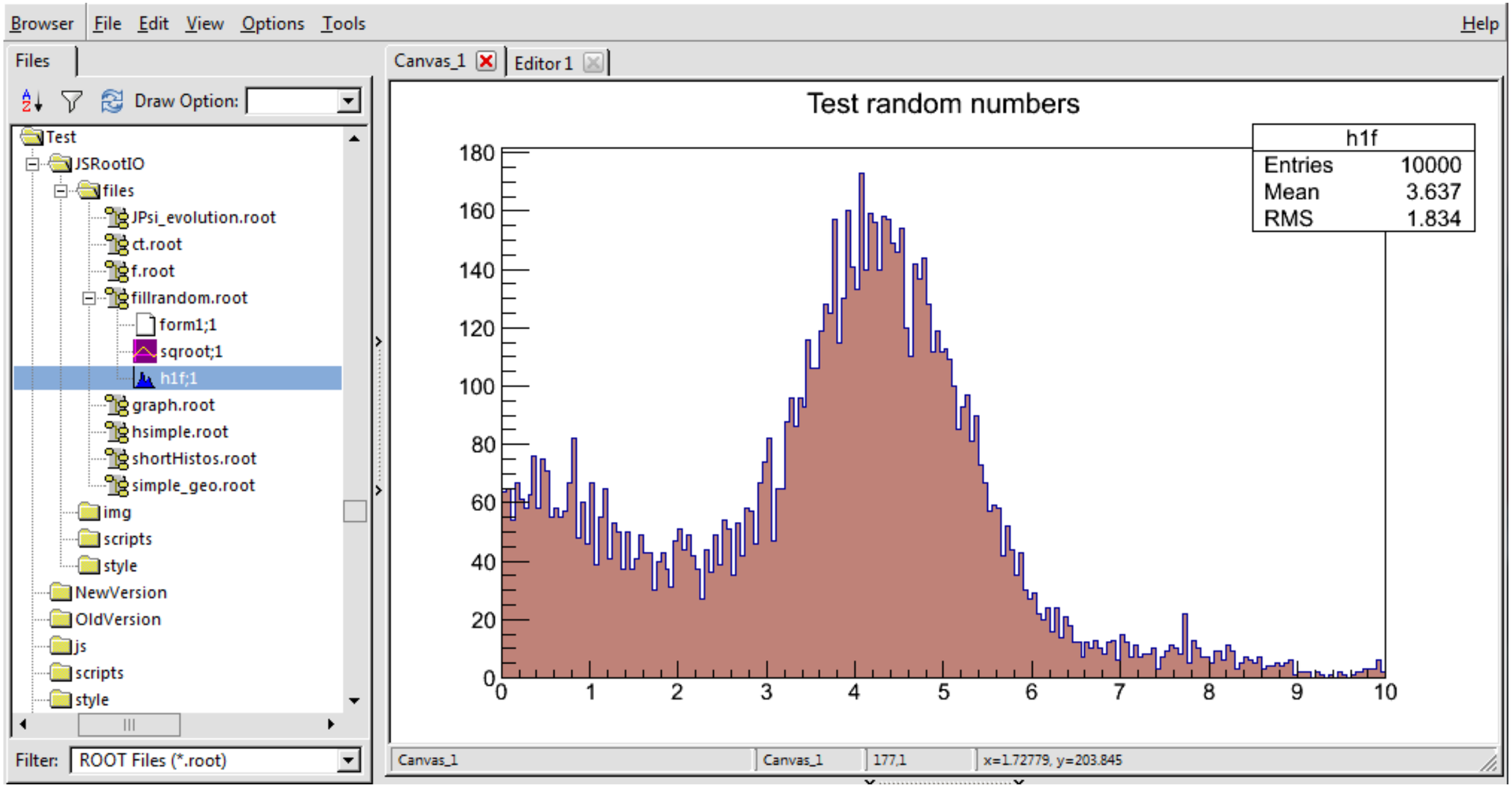
Reading objects, actual status

- One very nice feature of JavaScript is the possibility to dynamically (at runtime) create classes
- Allows to potentially read any object from a ROOT file, as soon as we can read the streamer info of its class

Graphics

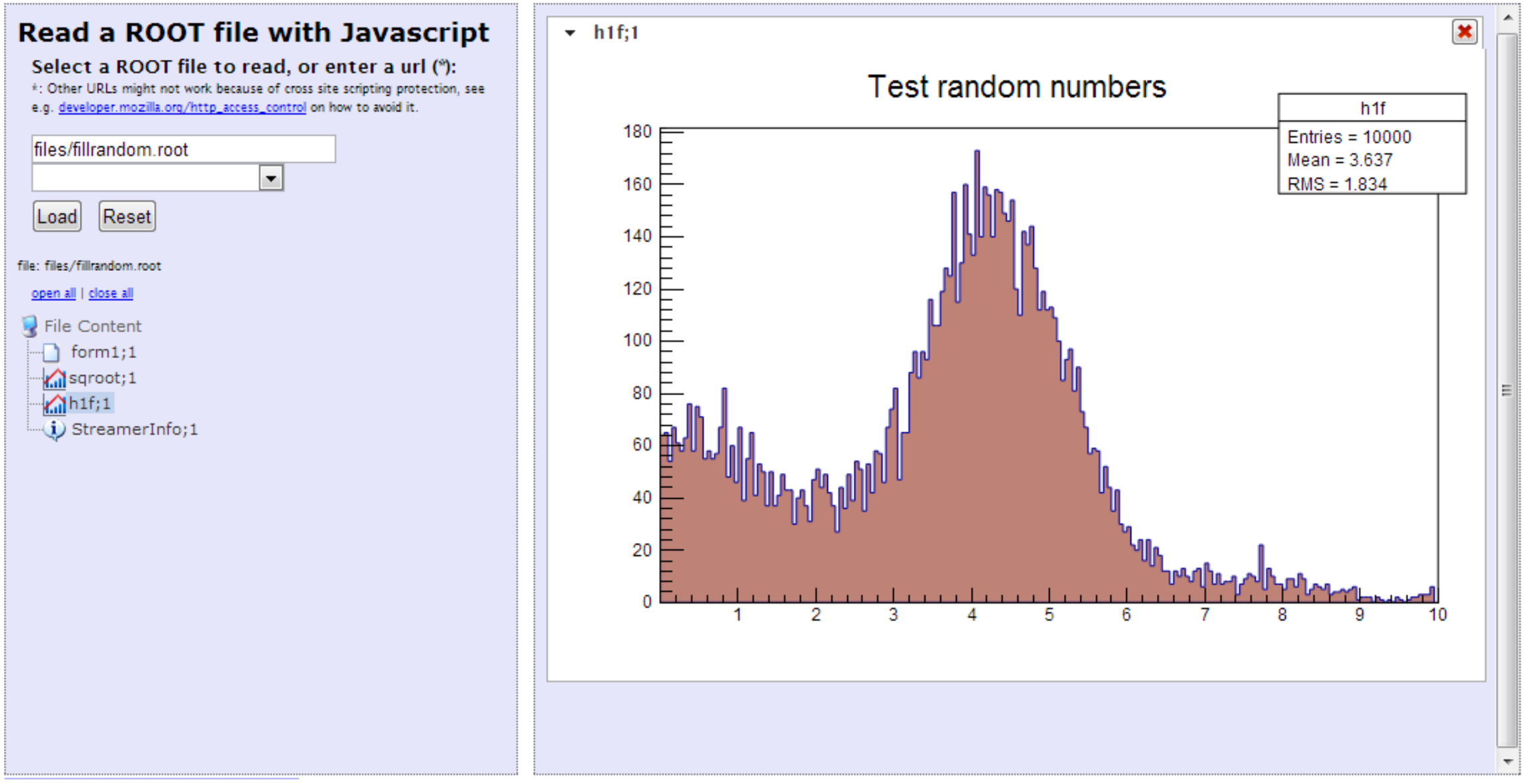
- A JavaScript library (d3.js) is used to display the 1D & 2D histograms and graphs (http://d3js.org/), and is released under the BSD License
- Another library (three.js) is used for 3D graphics (released under the MIT License)
- 3D graphics uses WebGL technology when available (browser and platform dependent)

Displaying objects



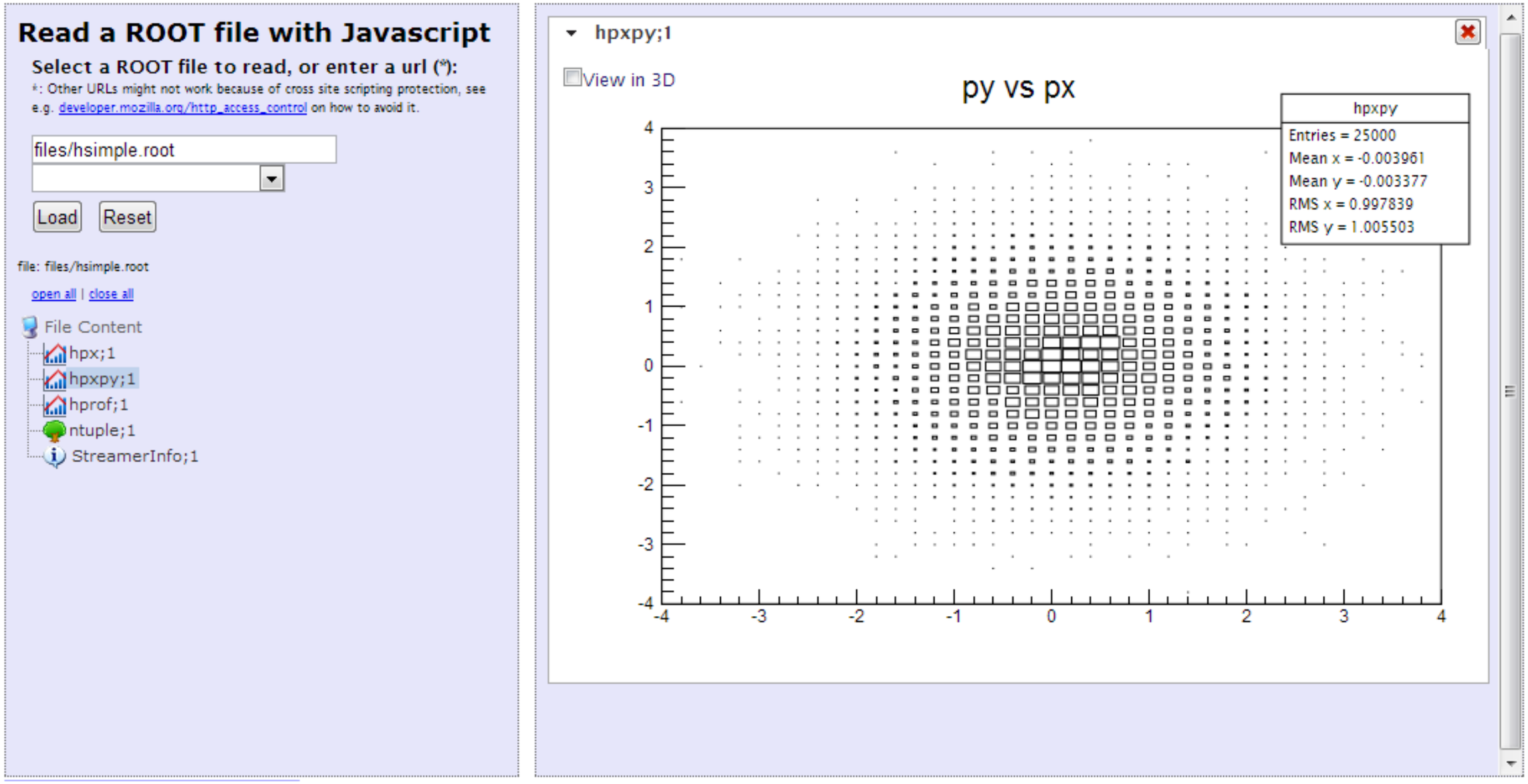
Traditional visualization of a local ROOT file in the ROOT browser

Displaying objects (cont.)



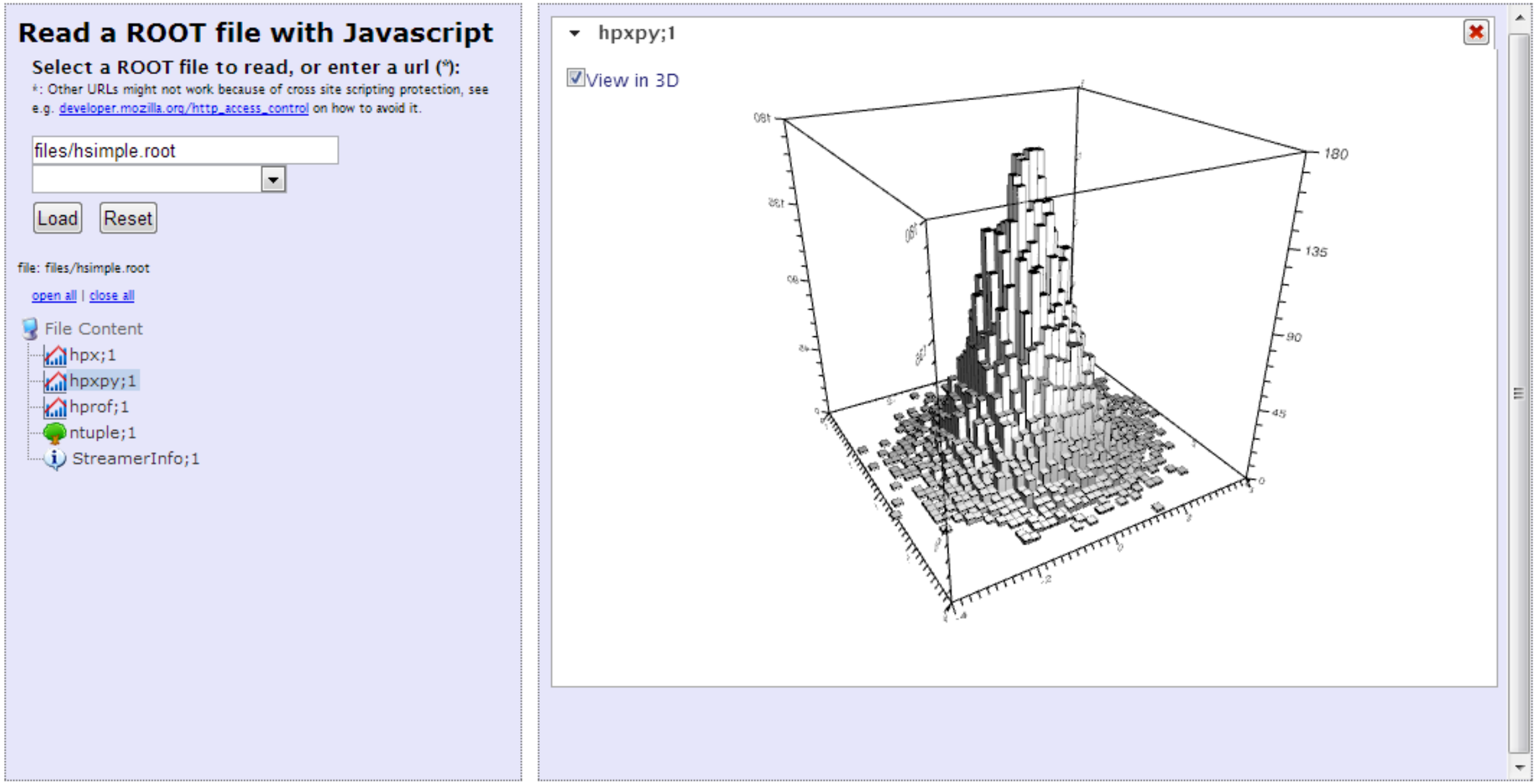
JSROOTIO.js visualization of identical histogram

Displaying objects (cont.)



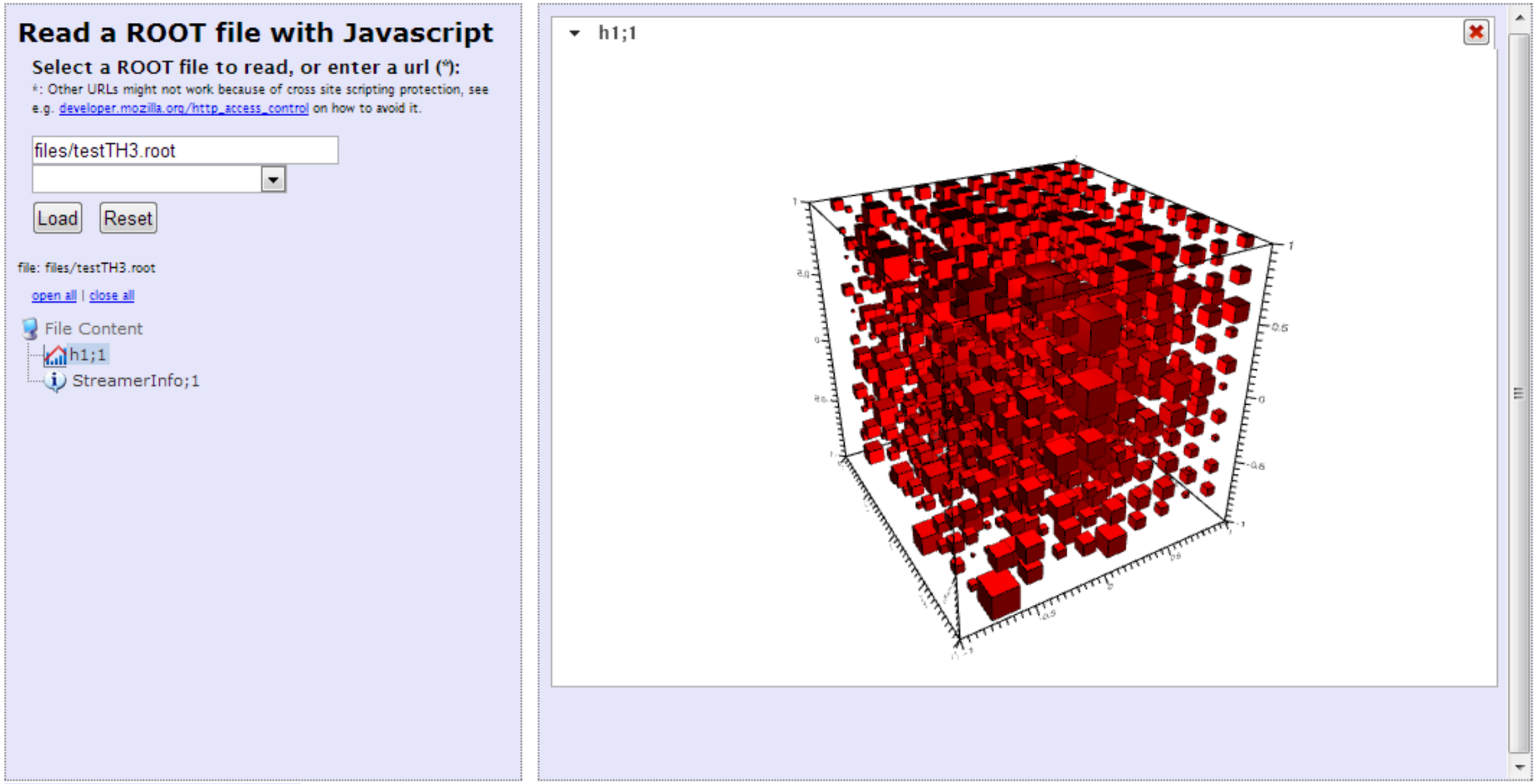
Displaying a TH2, as “BOX” plot (default)

Displaying objects (cont.)



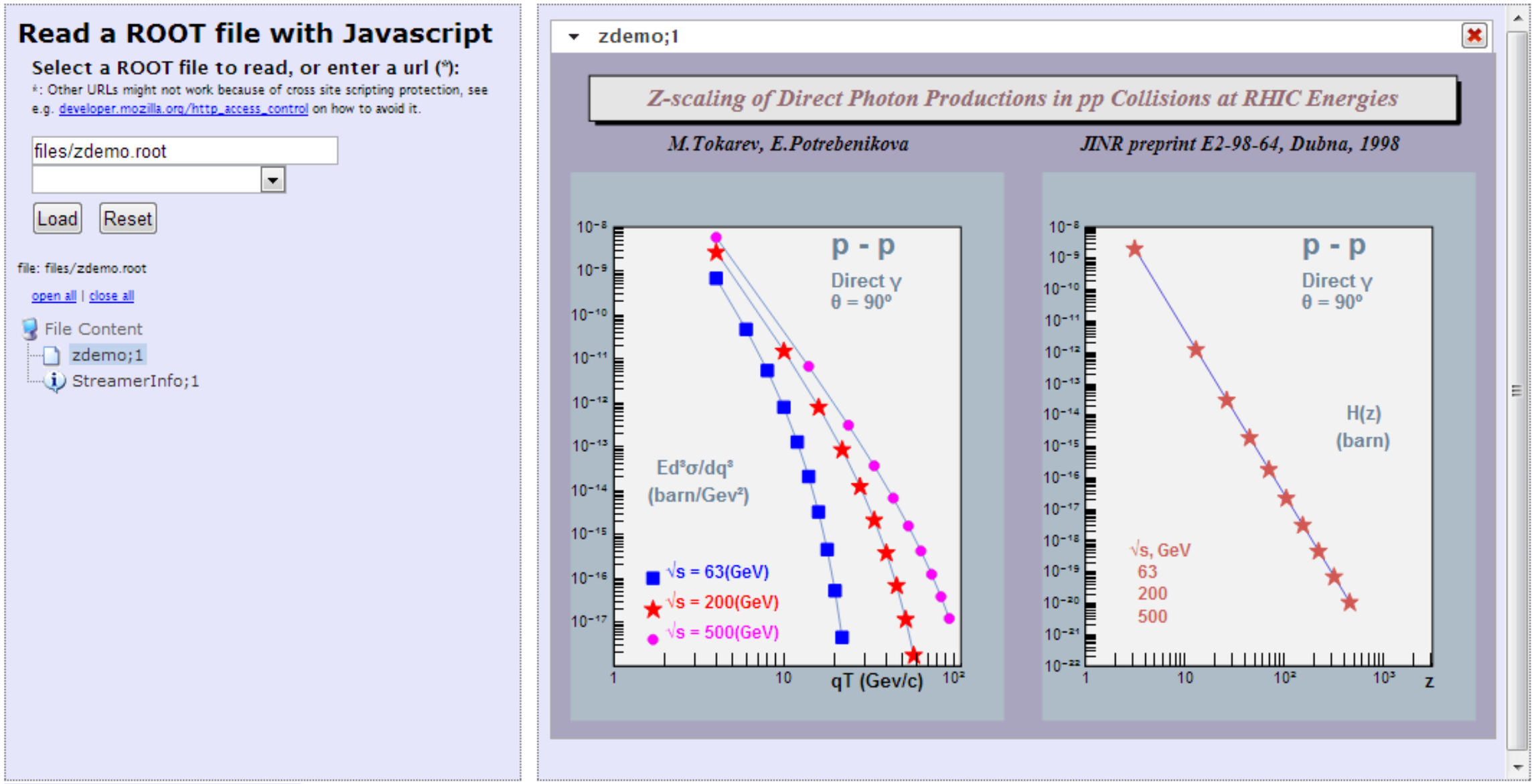
Displaying the same TH2, as “LEGO” plot

Displaying objects (cont.)



Displaying a simple TH3

Displaying objects (cont.)



Displaying the content of a TCanvas

How to use it?

- Simply copy the ROOT file(s) anywhere on the web
- Create a simple html page next to the files
  - Only two lines have to be added in the <head>

```
<head>
   <title>Read a ROOT file in JavaScript (Demonstration)</title>
   <link rel="stylesheet" type="text/css" href="http://root.cern.ch/js/style/JSRootInterface.css" />
   <script type="text/javascript" src="http://root.cern.ch/js/scripts/JSRootInterface.js"></script>
</head>
```

  - Including css and js directly from the root web site keeps you up to date with the latest version

Simple html Example

- And a few lines in the <body>. Here is a complete example:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
   <head>
      <title>Read a ROOT file in Javascript (Demonstration)</title>
      <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
      <link rel="stylesheet" type="text/css" href="http://root.cern.ch/js/style/JSRootInterface.css" />
      <script type="text/javascript" src="http://root.cern.ch/js/scripts/JSRootInterface.js"></script>
   </head>
   <body onload="BuildSimpleGUI()">
      <div id="simpleGUI" files="file_1.root;file_2.root;file_n.root;"></div>
   </body>
</html>
```

JavaScript API

- Offering a very simple API:

```
Var f = new JSROOTIO.RootFile(url);
var histo = f.ReadHistogram(histo_name);
if (typeof(histo) != 'undefined')
   displayHistogram(histo);
```

- But could be internally complex:

```
clRef = streamerInfo.ReadClass(str, o);
histo = eval('new JSROOTIO.' + clRef['name'] + '()');
if (typeof histo != 'undefined' &&
    typeof histo.Streamer == 'function')
   histo.Streamer(str, o);
```

Status, availability, coming next

- Doesn’t work on Android prior to version 4.0 (doesn't allow byte range HTTP requests)
- The source code is available in svn: http://root.cern.ch/svn/root/trunk/js/JSRootIO
- Remaining tasks
  - Finalize the automatic streaming
  - Allow to use custom streamers (partially done)
  - Add missing drawing options
  - Add graphical feedback (e.g. tooltips)
  - Add Latex support

Conclusion

The code is in a very good shape, thanks to early users who gave very valuable feedback

Feel free to try and to send feedback & requests