JavaScript ROOT

https://root.cern.ch/js/

Bertrand Bellenot (CERN)
Sergey Linev (GSI)
JSRootIO project

- Original project from Bertrand Bellenot
  - was presented at the ROOT workshop 2013

- ROOT I/O in JavaScript
  - reading ROOT objects using streamer infos
    - many exceptions due to custom streamers
  - source for the project name - JSRootIO

- Graphic with d3.js, three.js and jQuery.js

- Navigating the ROOT files content and displaying objects in modern web browsers
JSRootIO screenshots
JSRootIO and http server

- Could one use JSRootIO with online ROOT application?

- In general yes, but many hidden problems and caveats:
  - difficulty with fixed HTML design
  - lack of objects update
  - flexible API was missing

- There was an intermediate solution with many workarounds until a decision was taken to redesign JSRootIO completely
JavaScript ROOT

• Preserve old functionality, including look-and-feel

• Redesign was focused on:
  o modularity
  o clear API
  o more interactive features
  o more supported classes
  o support of user classes

• Project was renamed to **JSROOT**
  o binary ROOT files reading is an optional part of the project
Main features

JavaScript ROOT provides:

• Objects reading from binary and JSON ROOT files
• Display for popular ROOT classes in web browsers
• Flexible API for usage in other projects
How to use JSROOT?

• As before, for interactive browsing of ROOT files
  o open JSROOT web page https://root.cern.ch/js/latest/
  o load file(s) from web
  o show content of the files
  o display objects from the files
User interface

- file loading
- layout selection
- file content

py vs px

- hpxpy
  - Entries: 75000
  - Mean x: -0.001335
  - Mean y: -4.604e-4
  - RMS x: 1.002
  - RMS y: 1.001
Different layouts

grid

collapsible

tabs

simple
Context menu with draw options
Context menu for drawn object
Informative tooltips
Intuitive zooming

This is the px distribution

Entries 75000
Mean -0.001296
RMS 1.002

This is the px distribution

Entries 75000
Mean -0.01855
RMS 0.7641
How to share results?

• Before
  o create and send PNG image (static)
  o or create and send ROOT file with canvas (interactive)
    • one requires ROOT installed everywhere

• With JSROOT
  o copy your ROOT file on web server and send link to the file
  o open main page https://root.cern.ch/js/latest/
  o enter file name (like https://root.cern.ch/js/files/hsimple.root)
  o find and draw histogram or canvas

• Same actions repeat many times again 😞
  o interactive not always mean better
  o are there alternatives?

• Solution - use JSROOT UI with URL parameters!
URL parameters in JSROOT

- `file(s)` – name of file(s) to open
- `json` – name of json file to open
- `item(s)` – item name(s) to display
- `opt(s)` – drawing option for the item(s)
- `layout` – layout for drawings like grid or tabs
- `nobrowser` – do not display objects hierarchy
- `mathjax` – enable usage of MathJax.js
- `interactive` – enable/disable interactive features
- `load` – name of extra JavaScript to load
- `optimize` – drawing optimization (0: off, 1: large histos, 2: always)
URL: open file

https://root.cern.ch/js/latest/?file=../files/graph.root
URL: display canvas from file

https://root.cern.ch/js/latest/?file=../files/graph.root&item=exclusion;1&nobrowser
URL: display several items

https://root.cern.ch/js/latest/?file=../files/hsimple.root&layout=grid2x2&items=[hpx;1,hpxpy;1,hprof;1,StreamerInfo]&opts=[hist,colz,e1,any]

URL: display histograms from different files

https://root.cern.ch/js/latest/?layout=grid1x3&path=../files/&files=[temp44.root,temp35.root,temp28.root]&items=[temp44.root/Histograms/TDC_C100/Ch3/TDC_C100_Ch3_RisingRef;1,temp35.root/_same_,temp28.root/_same_]&opts=[autozoom,autozoom,autozoom]

http://bit.ly/1L5cvyJ
I/O improvements

• Make logic closer to original ROOT I/O
  o introduce JSROOT.TBuffer class
  o always use checksum to verify content

• Handle all custom streamers in central place
  o all kind of ROOT collections
  o TCanvas, TObjString, TStreamer... classes
  o make it easier to support user classes with custom streamers

• support ROOT4, ROOT5 and ROOT6 binary files

• support files reading from other web server
  o CORS headers should be enabled on the server
  o one can read files from local file system

• I/O fully independent from graphics
  o vice versa is also true
Graphics improvements

- Full code reorganization
- Introduce painter classes
  - somehow similar to original ROOT
- Make several SVG layers
  - axis, main drawing, labels
  - easier to overlap objects
- Comfort zooming and stat box update
- Context menu for additional functionality
- Significant performance increase
- Use of MathJax.js for equation drawings
Supported ROOT classes

- histograms:
  - TH1, TH2, TH3, TProfile

- graphs:
  - TGraph, TCutG, TGraphErrors, TGraphAssymErrors, TGraphBentErrors

- superposition:
  - THStack, TMultiGraph

- functions:
  - TF1

- text:
  - TLatex, TMathText, TPaveText, TPaveStats, TPaveLabel

- containers:
  - TCanvas, TPad
Modularity

- Code divided on several modules
  - core, 2d, 3d, io, gui

- Modules loaded when required
  - in simple case only three JSROOT scripts are loaded instead of 10 before
  - could be specified when loading central JSROOT script

```html
<script type="text/javascript"
```

- One could use require.js (optional)
  - example [https://root.cern.ch/js/latest/demo/exampleRequire.htm](https://root.cern.ch/js/latest/demo/exampleRequire.htm)

- Minified version of scripts are provided
Use in other HTML pages

• Simplest solution - `<iframe>` tag

```html
...<iframe width="800" height="500"
item=hp;1&nobrowser">
`/iframe`
</iframe>
...
```

• Not the first choice when many objects should be shown on the same page
Use in other HTML pages

• Load required functionality:

  
  <script type="text/javascript"

• Provide place for drawing object:

  
  <div id="drawing" style="width:800px; height:600px"></div>

• Retrieve object and call:

  
  JSROOT.draw("drawing", obj, "colz");
var filename = "https://root.cern.ch/js/files/hsimple.root";
new JSROOT.TFile(filename, function(file) {
    file.ReadObject("hpxpy;1", function(obj) {
        JSROOT.draw("drawing", obj, "colz");
    });
});

• See https://root.cern.ch/js/latest/demo/example_file.htm
Display object from JSON file

- **TBufferJSON** can create JSON representation
  - no need for binary ROOT I/O in JavaScript
  - more details in the THttpServer presentation on Friday

```javascript
JSROOT.NewRequest("hpx.json", "object", function(obj) {
    JSROOT.draw("drawing", obj, "hist");
}).send();
```

- See [https://root.cern.ch/js/latest/demo/example_json.htm](https://root.cern.ch/js/latest/demo/example_json.htm)
var cnt = 0;
setInterval(updateGUI, 2000);
...
function updateGUI() {
    var addr = "root" + (cnt++ % 20) + ".json";
    JSROOT.NewRequest(addr, "object", function(histo) {
        JSROOT.redraw("drawing", obj, "hist");
    }).send();
}

- See https://root.cern.ch/js/latest/demo/demo.htm
CERNBox integration

CERNBox provides a functionality analogous to Dropbox™ or similar system, and is managed by CERN IT department (http://cernbox.web.cern.ch). It now integrates JSROOT, allowing to display ROOT files content.
Support of user classes

• One needs to implement and register a drawing function
• Load the script together with JSROOT
• If necessary, provide a custom streamer for it

• Example with TEllipse class
  o JavaScript code (~70 lines)
    • http://jsroot.gsi.de/dev/demo/ellipse.js
  o Canvas from ROOT reference
    • http://jsroot.gsi.de/dev/index.htm?file=../files/ellipse.root&item=c1;1&load=demo/ellipse.js

• More examples in go4 framework
  o see Joern talk on Friday
TEllipse example

Examples of Ellipses

ROOT canvas

Examples of Ellipses

JSROOT canvas
Useful links

• Developers repository
  o [https://github.com/linev/jsroot](https://github.com/linev/jsroot)

• Latest stable version in ROOT
  o $ROOTSYS/etc/http

• All versions with documentation and examples:
  o [https://root.cern.ch/js/](https://root.cern.ch/js/)
  o [http://jsroot.gsi.de/](http://jsroot.gsi.de/)