

JavaScript GUI

JSROOT reloaded

Basic ideas

- The GUI (widgets) is on the client side (HTML, JavaScript, ...).
- The GUI action triggers either a JavaScript function call (on the client side), or pass a formatted C++ method to be called on the server side.
- On the server side (C++) the ROOT THttpServer class is expected to handle the events.
- Users' GUI could be created using any web GUI toolkit

JavaScript GUI libraries

- Benefits:
 - Large user communities
- Strategy:
 - Select one and use it internally
 - Provide basic support for any GUI library
 - Dedicated add-ons for JSROOT/THttpServer
 - Avoid strong bounding with single one
 - Give complete freedom to the users to use their favourite one

Complex GUI

We need to build complex GUI, like editors and Fit Panel, from the C++ objects (classes) information and define the method and arguments to be processed by the server side from its class members or Getters and Setters. This is very similar to what is currently done in C++.

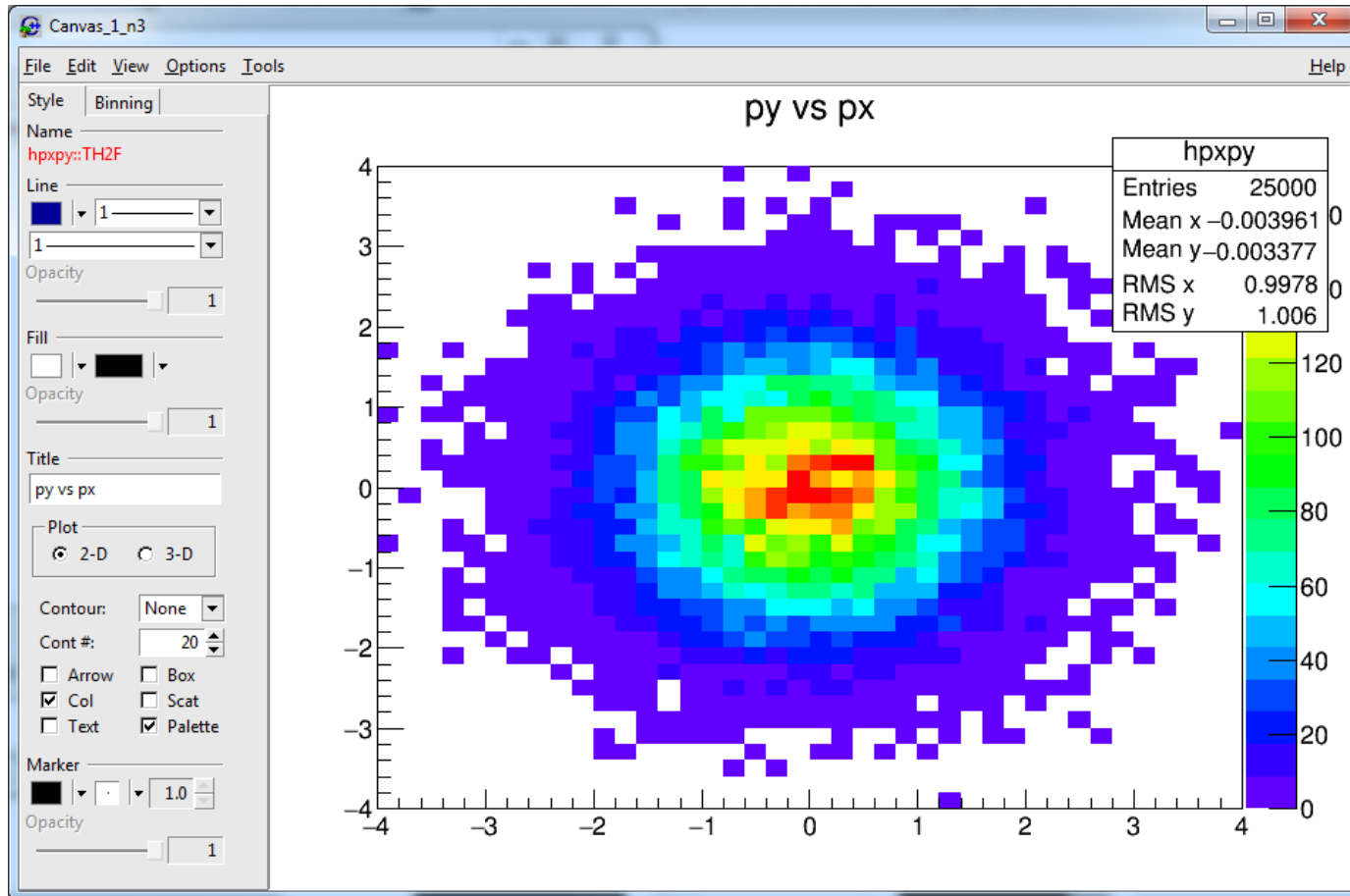
Graphical Editors

Objects like histograms, graphs, axis etc.. displayed in the canvas need local (on the JavaScript side) editors to change their attributes on the client side. Then a validation of the changes will trigger the update to the server side

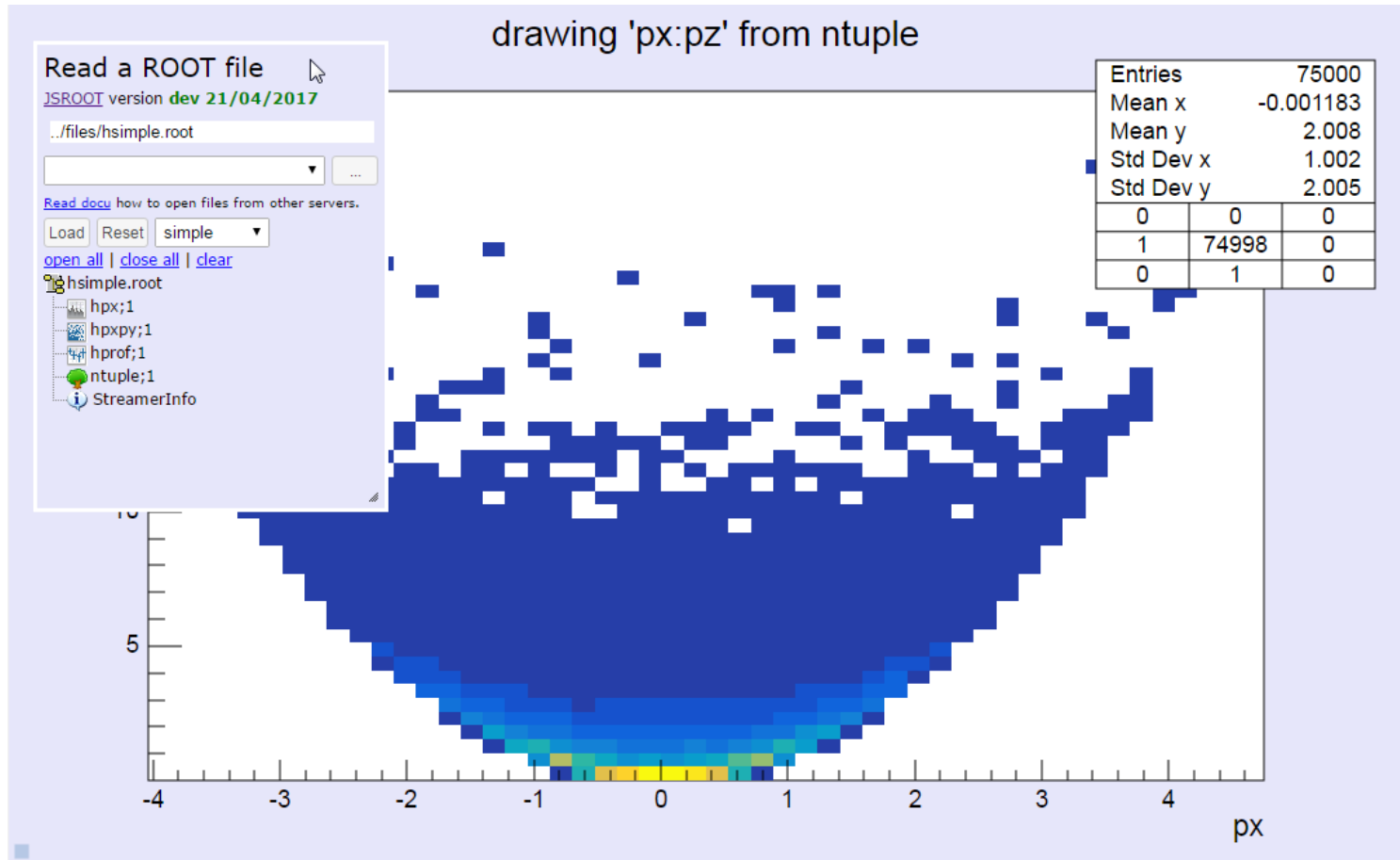
Possible options for the placement of the editors:

- On the left space (as in the browser)
- Floating, like [this example](#)
- Overlay (something like [this](#))

Left side (current layout)



Floating example



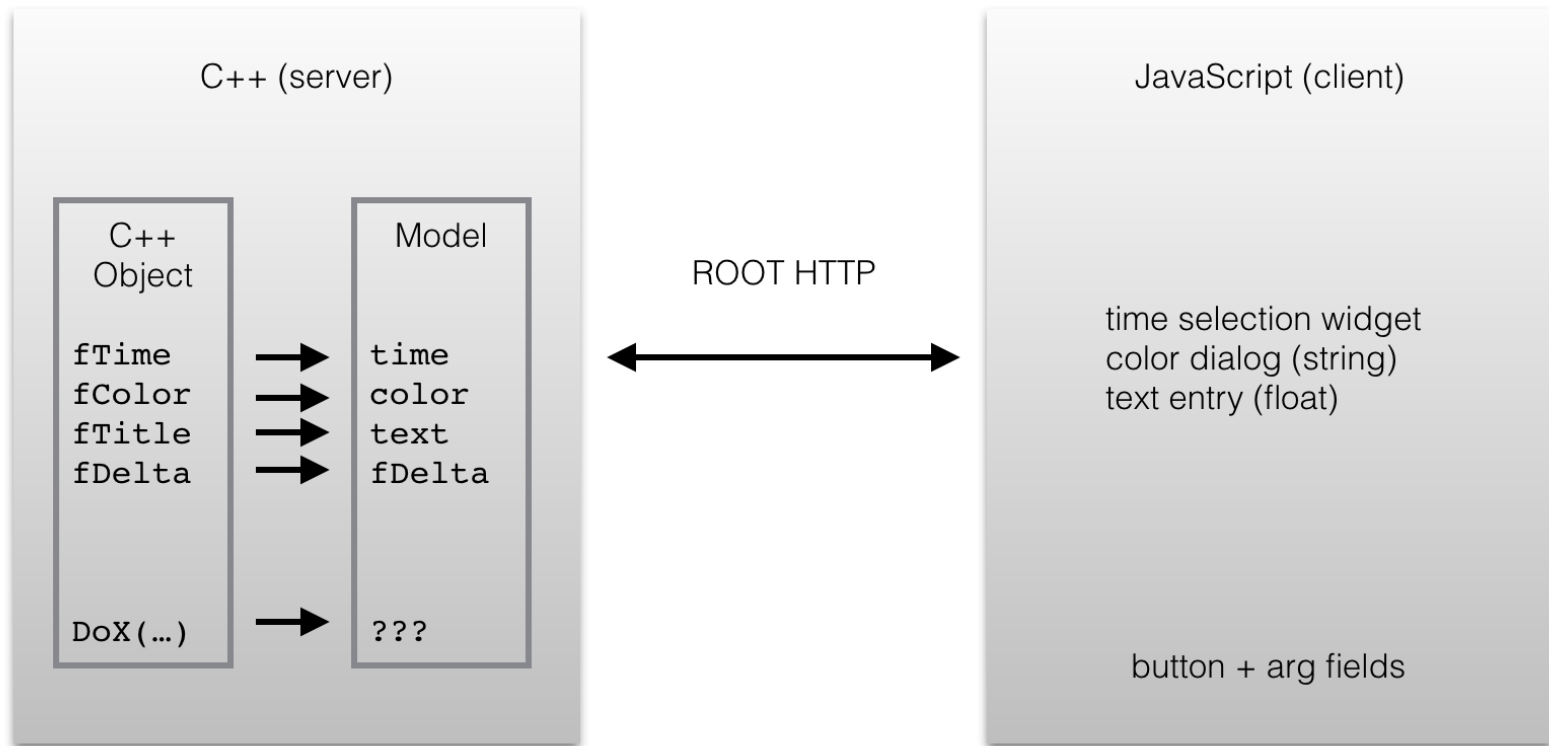
Overlay example

The image shows the Slidebean design tool interface. At the top, there is a navigation bar with the Slidebean logo and links for DASHBOARD, EXPLORE, and BLOG. On the right side of the navigation bar, there are icons for a gallery, a heart, a '+ New' button, and a user profile icon labeled 'SD'. Below the navigation bar, there is a secondary menu with 'SETTINGS', 'CONTENT', 'DESIGN' (which is highlighted with an orange underline), 'PRESENT', and 'INSIGHTS'. The main area of the interface is a slide editor. On the left, there is a large yellow number '2'. The central part of the editor shows a slide with a background image of a group of men in suits. The slide contains a quote: "The people who are crazy enough to think they can change the world are the ones who do." Below the quote is a white box with the text "Think Different". At the bottom of the slide, there is a black box with white text that reads: "go ahead and jump to the design section. In here you can change things like the". On the right side of the slide editor, there is a vertical sidebar with three sections: 'TEMPLATE' (showing a grid of template icons, with 'BOULEVARD' selected and highlighted in orange), 'COLORS' (showing a color palette with blue, black, and white swatches, labeled 'LUMBER'), and 'FONT' (showing a font preview with 'Abc' and the name 'LATO'). At the bottom of the interface, there is a video player control bar with a play button, a progress bar showing '1:00 / 1:25', an 'OPTIONS' button, and icons for closed captions (CC), HD, and full screen.

User's Defined GUI

- To implement their own GUI, users should:
 - Define the interface (model) from their C++ object to be implemented by the JavaScript GUI
 - Register the object in the http server, to make it visible/accessible on the client side
 - Create their GUI using the object's' methods/types

Interface (model) Definition



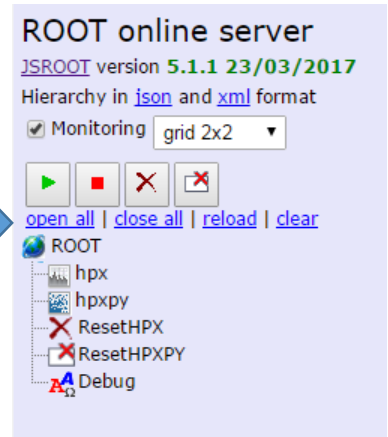
Current Status

- The TRootSniffer class gives full description of the ROOT objects hierarchy used with THttpServer
- It is already possible to add objects and actions to THttpServer:





```
// register histograms
serv->Register("/", hpx);

// register simple start/stop commands
serv->RegisterCommand("/Start", "bFillHist=kTRUE;",
                    "button;rootsys/icons/ed_execute.png");

// register commands, invoking object methods
serv->RegisterCommand("/ResetHPX", "/hpx/->Reset()",
                    "button;rootsys/icons/ed_delete.png");
```



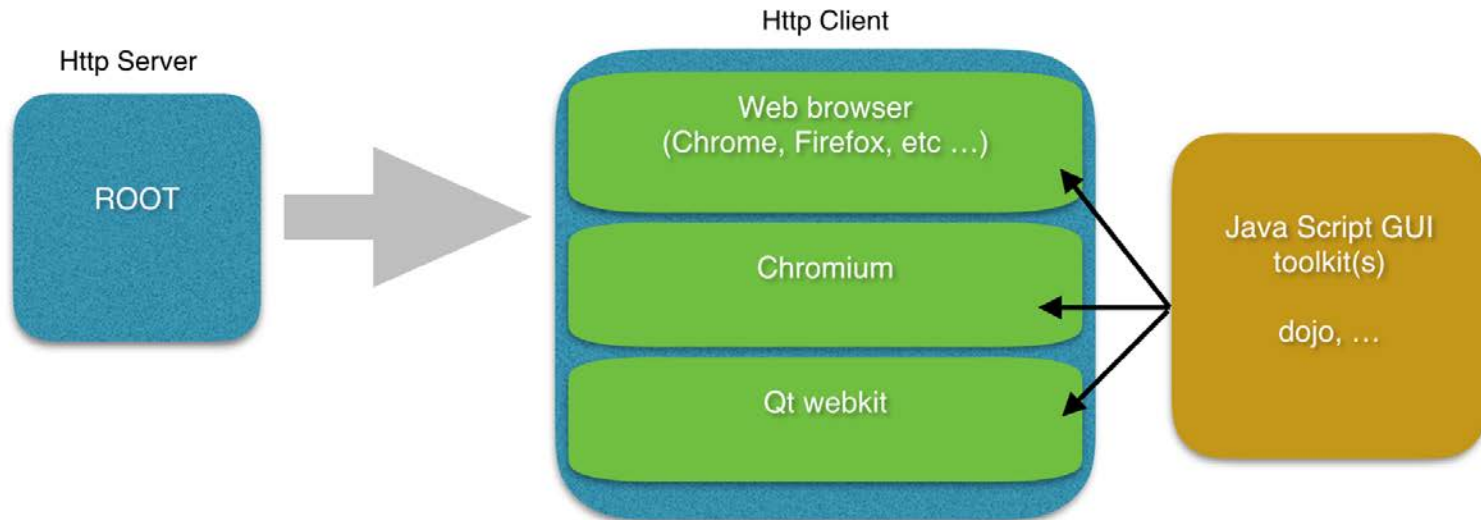
ROOT online server
JSROOT version 5.1.1 23/03/2017
Hierarchy in [json](#) and [xml](#) format
 Monitoring

[open all](#) | [close all](#) | [reload](#) | [clear](#)

- ROOT
 - hpx
 - hpxpy
 - ResetHPX
 - ResetHPXPY
 - Debug

Client-Server architecture



HTTP Client

- Any Web browser like Firefox, Chrome, etc...
Allowing to render and interact with the graphics locally or remotely
- Local client allowing to render and interact with the graphics locally only. Examples of such tool are libchrome (from the Chromium project) and the Qt Webkit
- libchrome need investigations
- Some preliminary tests have been done with WebKit

WebKit - Preliminary Tests

- Qt4 and Qt5 implements QWebView widget, based on WebKit.
- Such widget can be integrated into any qt-based GUI as any other normal QWidget.
- Some preliminary tests with the "fancybrowser" example from both qt4 and qt5 show that JSROOT graphics in general works! One could display SVG and WebGL graphics smoothly.

JavaScript GUI libraries

- There are many of them
 - jqWidgets <http://www.jqwidgets.com> (Creative Commons Attribution-NonCommercial 3.0)
 - Dijit (based on dojo) <https://dojotoolkit.org> (New BSD or Academic Free License version 2.1)
 - Webix <http://webix.com> (GPL v3)
 - OpenUI5 <http://openui5.org> (Apache-2.0)
 - ...
- See also
 - <http://stackoverflow.com/questions/200284>
 - https://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks

jQWidgets example

The screenshot displays a web development IDE interface with the following components:

- Toolbox:** Located on the left, containing a 'Help' button.
- Document Tabs:** At the top, showing 'Document 1' and 'Document 2'.
- Main Canvas:** The central area is divided into two views:
 - Left View:** A 3D perspective view of a complex, multi-colored geometric object (purple, blue, green, red) with a central cylindrical component.
 - Right View:** A 2D top-down view of a circular chart or radar plot with multiple overlapping colored lines (purple, blue, green, red) and a central point.
- Solution Explorer:** Located on the right side, showing a project structure:
 - Project
 - css
 - jqx.base.css
 - jqx.energyblue.css
 - jqx.orange.css
 - scripts
 - index.htm
- Output Window:** An orange-bordered window in the bottom right corner displaying the following text:

```
Themes installation complete.

List of installed stylesheet files. Include at least one stylesheet
Theme file and the images folder:

• styles/jqx.base.css: Stylesheet for the base Theme. The
jqx.base.css file should be always included in your project.
• styles/jqx.arctic.css: Stylesheet for the Arctic Theme
• styles/jqx.web.css: Stylesheet for the Web Theme
• styles/jqx.bootstrap.css: Stylesheet for the Bootstrap Theme
• styles/jqx.classic.css: Stylesheet for the Classic Theme
• styles/jqx.darkblue.css: Stylesheet for the DarkBlue Theme
• styles/jqx.energyblue.css: Stylesheet for the EnergyBlue
Theme
```
- Error List:** Located at the bottom left, showing 'Error List' and 'List of errors'.

Dijit Example

The image shows a web-based IDE interface. On the left is a file explorer with a tree view containing folders like 'Classes', 'Colors', 'MapFiles', 'Sockets', 'Canvases', 'Styles', 'Functions', 'Tasks', 'Geometries', 'Browsers', 'Specials', 'Cleanups', 'StreamerInfo', 'SecContexts', 'PROOF Sessions', 'ROOT Memory', 'ROOT Files', and 'Users'. The 'ROOT Files' folder is expanded, showing 'hsimple.root'. A menu is open over the 'Options' tab, listing various settings such as 'Auto Resize Canvas', 'Move Opaque', 'Resize Opaque', 'Interrupt', 'Refresh', 'Pad Auto Exec', 'Statistics', 'Histogram Title', 'Fit Parameters', and 'Can Edit Histogram'. The 'Histogram Title' option is highlighted. The main editor area, titled 'Editor 1', contains C++ code for a histogram class. The code includes a template for dimensions and precision, a class definition for 'THist', and various utility types and functions.

```
template<int DIMENSIONS, class PRECISION,
        template <int D_, class P_, template <class P__> class S_> class... STAT>
class THist {
public:
    /// The type of the 'Detail::THistImplBase' of this histogram.
    using ImplBase_t
        = Detail::THistImplBase<Detail::THistData<DIMENSIONS, PRECISION,
            Detail::THistDataDefaultStorage, STAT...>>;

    /// The coordinates type: a 'DIMENSIONS'-dimensional 'std::array' of 'double'.
    using CoordArray_t = typename ImplBase_t::CoordArray_t;
    /// The type of weights
    using Weight_t = PRECISION;
    /// Pointer type to 'HistImpl_t::Fill', for faster access.
    using FillFunc_t = typename ImplBase_t::FillFunc_t;
    /// Range.
    using AxisRange_t = typename ImplBase_t::AxisIterRange_t;

    using const_iterator = Detail::THistBinIter<ImplBase_t>;

    /// Number of dimensions of the coordinates
    static constexpr int GetNDim() noexcept { return DIMENSIONS; }
```

Webix Example

The screenshot shows a web application interface with a top navigation bar containing three buttons: "Browser" (with a dropdown arrow), "Next Event", and "Previous Event". On the left side, there is a file browser panel with a tree structure:

- Home
 - ROOT Files
 - simple_alice.root
 - fitslicesy.root

The main content area is divided into three vertical panels:

- Event Display:** A large 3D visualization of a particle detector cross-section. It features a central cylindrical structure with various colored components (orange, green, purple). Numerous purple tracks and blue dots are overlaid on the detector, representing particle paths and interaction points.
- Canvas:** A panel that is currently empty.
- Editor:** A panel containing two sub-views. The top view is a circular cross-section of the detector with a complex network of purple and green tracks. The bottom view is a rectangular cross-section showing a similar track network. Both views include small control icons at the bottom left.

What Next

- Select a JavaScript GUI library
- Implement wrappers (C++ and JavaScript)
- Implement a JavaScript Tree Viewer