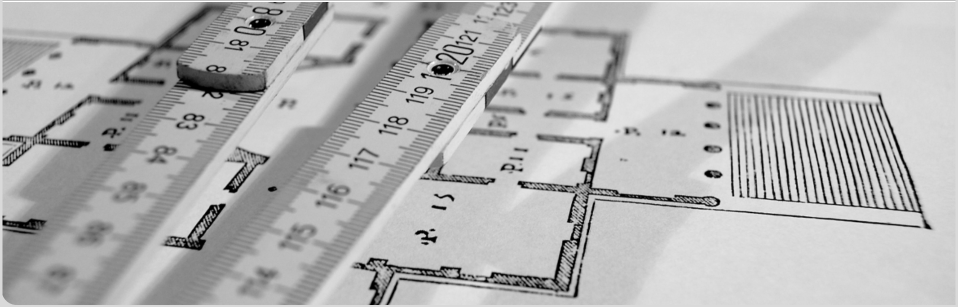
rootJS - Specification

PSE - Software Engineering Practice

C. Wolff, M. Früh, S. Rajgopal, C. Haas, J. Schwabe, T. Beffart | December 14, 2015

STEINBRUCH CENTER FOR COMPUTING



Outline/Gliederung

Purpose

Product usage

rootJS will be used to create web-applications that can:

- Expose processed data (that might otherwise be hard to access) and then visualize it locally
- Interact with data both stored somewhere accessible for the server or streamed via remote procedure call (RPC)
- Run on any platform that supports a browser

Audience

Most users of rootJS will be used to working in Linux and with web servers. At the very least, they will be able to install ROOT and also be proficient in programming languages like JavaScript and C++.

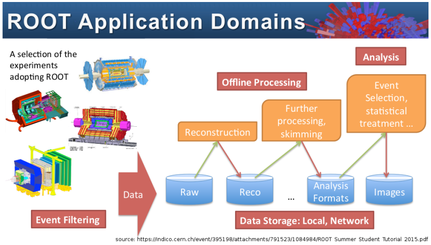
- Scientists (e.g. particle physicists)
- Researchers
- Web-developers interested in creating applications based on ROOT

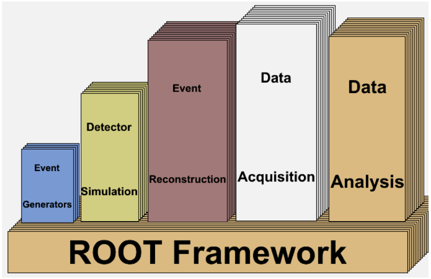
Operating conditions

- rootJS will be used on servers that run ROOT and have access to the required data sources.
- As ROOT 6 currently runs on Linux and OS X only, usage of the bindings is limited to those platforms.

ROOT

- process and visualize large amounts of scientific data (CERN)
- features a C++ interpreter (CLING) - i.e. used for rapid and efficient prototyping
- persistency mechanism for C++ objects


ROOT Application Domains
A selection of the experiments adopting ROOT
Analysis
Offline Processing
Event Selection, statistical treatment ...
Further processing, skimming
Reconstruction
Data
Raw
Reco
...
Analysis Formats
Images
Event Filtering
Data Storage: Local, Network
source: https://indico.cern.ch/event/395198/attachments/791523/1084984/ROOT Summer Student Tutorial 2015.pdf



Node.js

- open source runtime environment
  - develop server side web applications
  - act as a stand alone web server

Node.js

- open source runtime environment
    - develop server side web applications
    - act as a stand alone web server
- Google V8 engine to execute JavaScript code
- rootJS bindings realized as native Node.js module written in C++

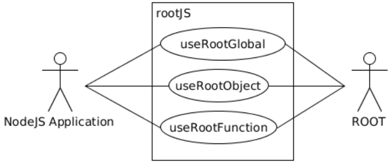
Hardware

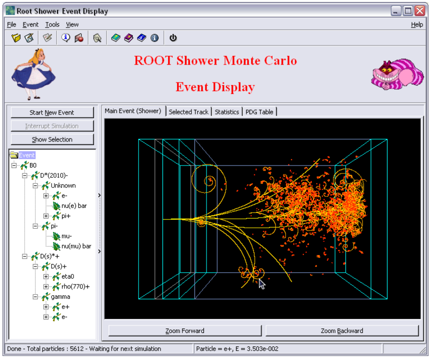
Product data

Product interface and functions

Scenario 1

Event Viewer





Done - Total particles : 5612 - Waiting for next simulation

Particle = e+, E = 3.503e-002

Initialization

- Expose all
  - Global variables
  - Global functions
  - Classes

Initialization

- Expose all
  - Global variables
  - Global functions
  - Classes
- Each are bound to corresponding proxy methods
- An object which members are the exposed features is beeing passed to node

Initialization

Karlsruhe Institute of Technology

- Expose all
  - Global variables
  - Global functions
  - Classes
- Each are bound to corresponding proxy methods
- An object which members are the exposed features is beeing passed to node

Names

- Functions and classes have the same name as in Root
- Global variables can be called using Get[Variable] and Set[Variable] methods

Call a feature

- All features in node are mapped to a proxy method that will be called

Call a feature

- All features in node are mapped to a proxy method that will be called
- The proxy method will eventually call a root function and pass the result to our ObjectFactory

Call a feature

- All features in node are mapped to a proxy method that will be called
- The proxy method will eventually call a root function and pass the result to our ObjectFactory
- By looking at the object type an corresponding v8::Handle will be generated and returned to node
  - If the result is an object this will be done recursively

Test Cases

References I