



rootJS - Specification

PSE - Software Engineering Practice

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

STEINBUCH CENTER FOR COMPUTING



Purpose



< □ > < 圖 > < 불 > < 불 > 분 | 표 의 의 ()

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

Product Environment Product data Product interface and functions

Use Cases System Model Test

Product usage

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

Product Environment Product data Product interface and functions Scenarios Use Cases System Model Test

Operating conditions

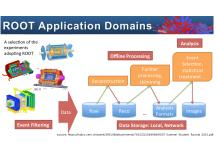


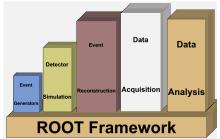
- rootJS will be used on servers that run ROOT and have access to the required data sources.
- As ROOT 6 currently runs on Linux and OS X only, usage of the bindings is limited to those platforms.

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





Scenarios



System Model Test

Node.js



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

December 14, 2015

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++

Use Cases System Model Test

Product Environment Product data Product interface and functions

Product usage

Hardware



< □ > < 圖 > < 불 > < 불 > 분 | 표 의 의 ()

8/16

December 14, 2015

Purpose Product usage Product Environment Product data Product interface and functions Scenarios Use Cases System Model Test

Product data

C. Wolff, M. Früh, S. Rajgopal, C. Haas, J. Schwabe, T. Beffart - rootJS





9/16

December 14, 2015

Product interface and functions





Scenario 1

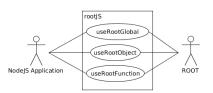


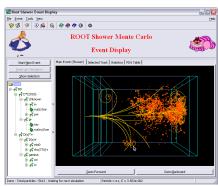
<ロ > ← □ > ← □ > ← □ > ← □ = ・ つ へ ○

Purpose Product usage Product Environment Product data Product interface and functions Scenarios Use Cases System Model Test

Event Viewer









Purpose

Product usage

Initialization



- Expose all
 - Global variables
 - Global functions
 - Classes



Initialization



- Expose all
 - Global variables
 - Global functions
 - Classes
- Each are bound to corresponding proxy methods

Product Environment Product data Product interface and functions

An object which members are the exposed features is beeing passed to node

Use Cases System Model Test

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

Names

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

Product Environment Product data Product interface and functions Scenarios



Product usage

Use Cases System Model Test

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

Product Environment Product data Product interface and functions Scenarios

If the result is an object this will be done recursively

Use Cases System Model Test

Test Cases

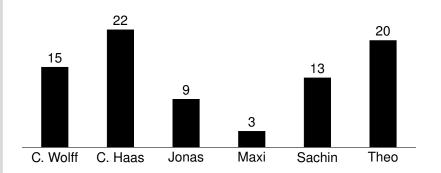


< □ > < 圖 > < 量 > < 量 > 至 |= り < ○

Purpose Product usage Product Environment Product data Product interface and functions Scenarios Use Cases System Model Test

Merges







December 14, 2015

References I

