



# rootJS - Specification

PSE - Software Engineering Practice

C. Wolff, M. Früh, S. Rajgopal, C. Haas, J. Schwabe, T. Beffart | December 15, 2015

#### STEINBUCH CENTER FOR COMPUTING



### About PSE



Praxis der Softwareentwicklung(PSE) = Software Engineering Practice

- Waterfall model
  - Planing/definition

Environment

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

Functional specification



# **Purpose**

Purpose



#### Node.js bindings for ROOT

- be able to write ROOT code in Node.js programs
- integrate ROOT into Node.js based web applications

Interface

Scenarios Use Cases

Test Cases

System Model

# **Required Criteria**



#### The bindings must

- work on Linux
- allow the user to interact with any ROOT class from the Node.js JavaScript interpreter
- accept C++ code for just-in-time compilation
- update dynamically following changes to C++ internals
- provide asynchronous wrappers for common I/O operations (i.e. file and tree access)

# **Optional Criteria**



#### The bindings should

- support the streaming of data in JavaScript Object Notation (JSON) format compatible with JavaScript ROOT
- implement a web server based on Node.js to mimic the function of the ROOT HTTP server

Scenarios Use Cases

work OS independent (i.e. support Mac OS X, Linux operating systems)

Test Cases

Environment

Purpose

# Limiting criteria



#### The bindings should not

add any extending functionality to the existing ROOT framework

Scenarios Use Cases

necessarily support previous/future ROOT versions

Interface

Data

Test Cases

Purpose

System Model

# 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



Purpose

### **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

Test Cases

# **Operating conditions**



rootJS will be used on servers that run ROOT and have access to the required data sources.

Scenarios Use Cases

As ROOT 6 currently runs on Linux and OS X only, usage of the bindings is limited to those platforms.

Test Cases

Purpose

### **ROOT**

Purpose

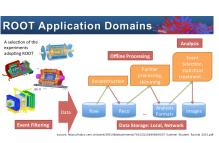


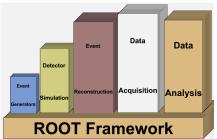
- process and visualize large amounts of scientific data (CERN)
- features a C++ interpreter (CLING) i.e. used for rapid and efficient prototyping

Scenarios

Use Cases

persistency mechanism for C++ objects





System Model



Test Cases

# 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





# 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**



- Task: encapsulation of ROOT objects and functions
  - → scanning ROOT structures during initialization
  - → encapsulating objects with heavily nested object structures

Scenarios Use Cases

→ introduce (proxy) object cache

System Model

Purpose

Test Cases

#### **Hardware**



- Task: encapsulation of ROOT objects and functions
  - ightarrow scanning ROOT structures during initialization
  - → encapsulating objects with heavily nested object structures
  - → introduce (proxy) object cache

⇒ generally negligible hardware requirements of the bindings themselves

Scenarios Use Cases



Test Cases

Purpose

#### Product data



### The following data will be stored by the rootJS bindings

- All ROOT classes and methods as they dynamically mapped to their JavaScript equivalents
- **ROOT** environment state
- Application context is derived from TApplication
- Map of v8::handles 2 identified by the address of ROOT objects

Scenarios Use Cases

Test Cases

Environment

Data

Purpose

### **Product interface**



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

Interface

Scenarios

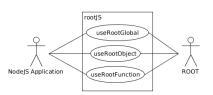
Use Cases

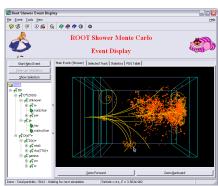
System Model

Test Cases Statistics

#### **Event Viewer**











Client application

API

ROOT framework

TROOT

Data



Test Cases

System Model

Scenarios

Use Cases

Interface



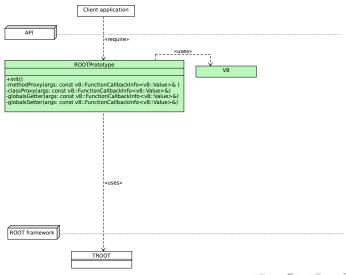




TROOT







Scenarios

Use Cases

System Model

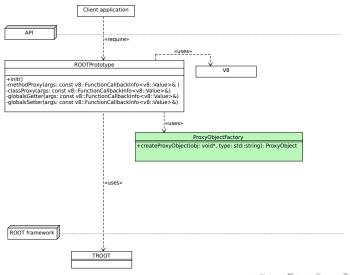
0000000000000000



Purpose

Test Cases





Scenarios

Interface

Use Cases

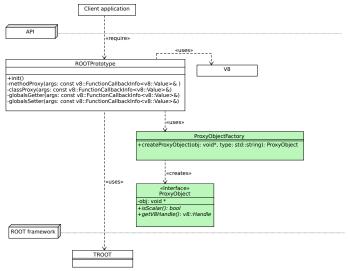
System Model



Test Cases

Purpose





Scenarios

Use Cases

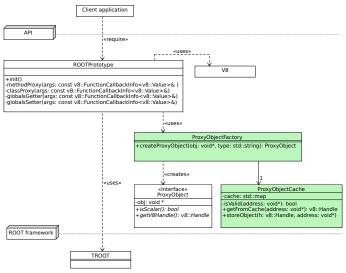
Interface

Test Cases

Purpose

System Model





Scenarios

Use Cases



Test Cases

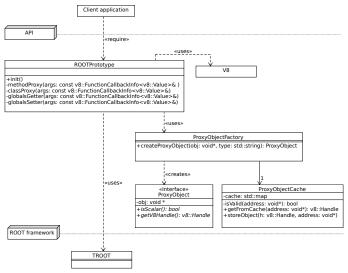
Data

**PSE** 

Purpose

System Model





Scenarios

Use Cases

Test Cases

Data

**PSE** 

Purpose

System Model

### 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



- 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**

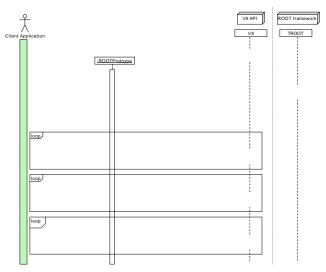
- Functions and classes have the same name as in Root
- Global variables can be called using Get[Variable] and Set[Variable] methods



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

Test Cases





Use Cases

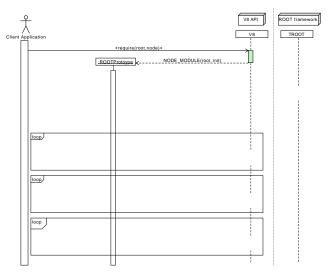
System Model

0000000000000000

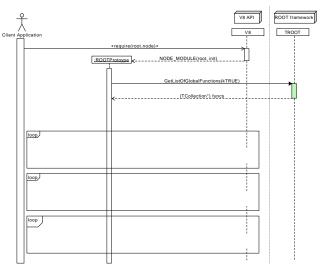
Test Cases

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

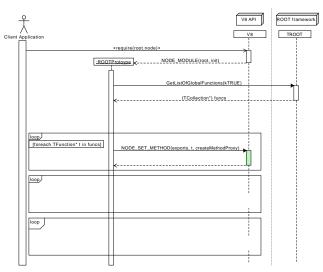




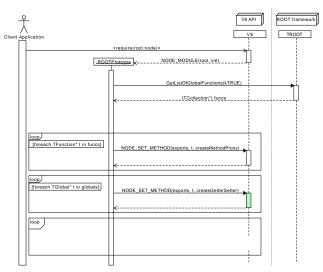






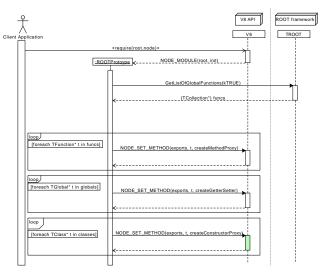






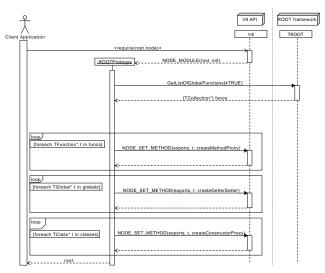












#### Call a feature



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



Test Cases

Data

Interface

Scenarios

Use Cases

System Model

00000000000000000

Purpose

#### Call a feature

Purpose



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

Use Cases

 The proxy method will eventually call a root function and pass the result to our ObjectFactory

Scenarios

Test Cases

#### 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



Purpose

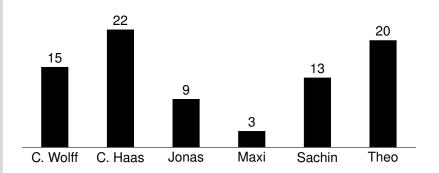
### **Test Cases**



# Merges

**PSE** 







Test Cases

Data

Interface

Scenarios

Use Cases

System Model

# **Punchcard**

Purpose

Usage





Data

Interface

Scenarios

Use Cases

System Model

#### References I



- CERN. ROOT application domains. Dec. 2015. URL: https://root.cern.ch/application-domains.
- CERN. ROOT Shower Event Display. Dec. 2015. URL: https://root.cern.ch/rootshower00png.
- exortech. v8 logo. Dec. 2015. URL: https://github.com/exortech/presentations/blob/master/promise\_of\_node/img/v8.png.
- Node.js logo. Dec. 2015. URL: https://nodejs.org/static/images/logos/nodejs-light.eps.
- Danilo Piparo and Olivier Couet. ROOT Tutorial for Summer Students. Dec. 2015. URL: https://indico.cern.ch/event/395198/attachments/

791523/1084984/ROOT\_Summer\_Student\_Tutorial\_2015.pdf.



## References II



Boris Vacher. npm logo. Dec. 2015. URL:

https://commons.wikimedia.org/wiki/File:Npm-logo.svg.