Karlsruhe Institute of Technology

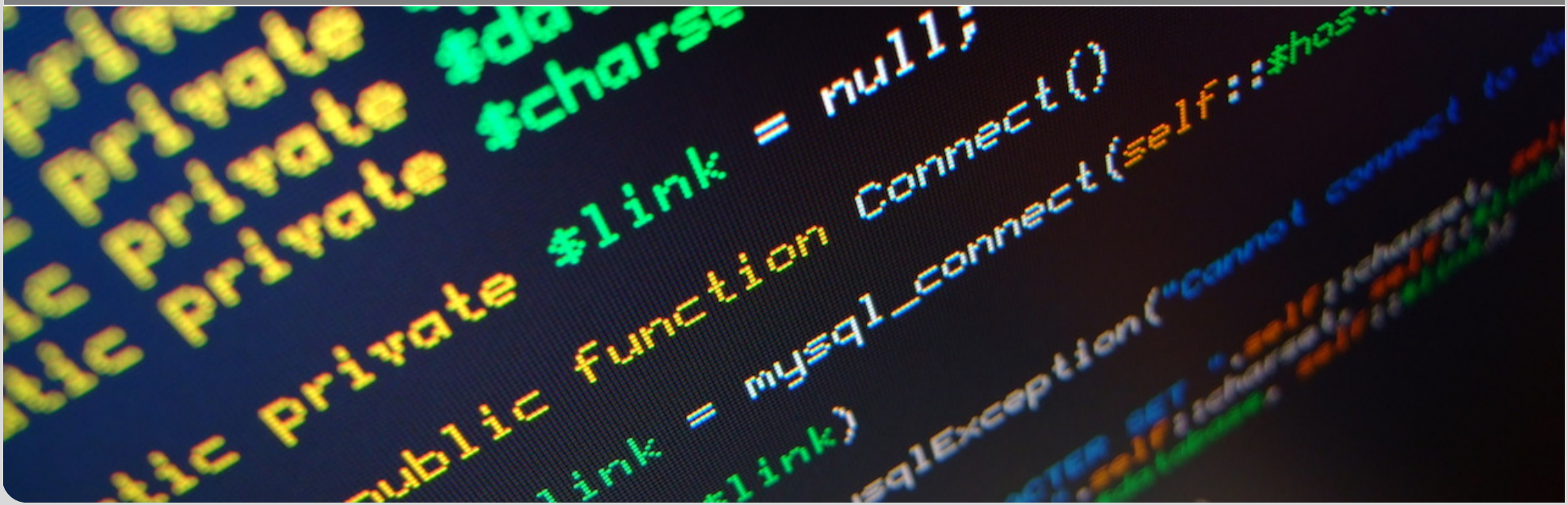
Steinbuch Centre for Computing

rootJS - results of the implementation phase

Software Engineering Practice - Winter Term 2015/16

C. Wolff, M. Früh, S. Rajgopal, C. Haas, J. Schwabe, T. Beffart | February 14, 2016

STEINBUCH CENTER FOR COMPUTING



Outline

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)

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)

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)

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)

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)

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) ✓

Limiting criteria

The bindings should not

- Add any extending functionality to the existing ROOT framework
- Necessarily support previous/future ROOT versions

Limiting criteria

The bindings should not

- Add any extending functionality to the existing ROOT framework ✓
- Necessarily support previous/future ROOT versions

Limiting criteria

The bindings should not

- Add any extending functionality to the existing ROOT framework ✓
- Necessarily support previous/future ROOT versions ✓

AsyncRunner

- We added a helper class for asynchronous operations, 'AsyncRunner'
- Uses libuv internally
- TThread doesn't allow communtication between node threads

MetaInfo

- We added a helper class to encapsulate differences between TGlobals and TDataMember
- TGlobal has a getAdress() method to get the absolute adress, TDataMember only getOffset()

Other changes

- Dynamic library loader: ROOT doesn't load enough libraries on its own
- removed get exports

Workflow

- Test-driven development
- We couldn't use Travis CI, the build of root always timed out
- So we used Jenkins instead
- Implement new features in branches first, then merge later

Statistics

- 4500+ LOC
- 300+ commits