



# 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

4 ロ ト 4 周 ト 4 E ト 4 E ト 9 Q C

### **Outline**



- Fullfilled criteria
  - Required criteria
- Changes in design
  - AsyncRunner
  - MetaInfo
  - Other changes
- Workflow
- Statistics



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

Statistics

2/9



#### 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)



#### 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)



#### 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)



#### 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)



#### 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)



#### 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) ✓

# **AsyncRunner**



- We added a helper class for asynchronous operations, 'AsyncRunner'
- Uses libuv internally
- TThread doesn't allow communication 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()
- Provide interface for return types



# 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**



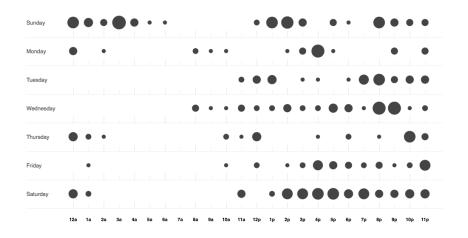
- 3200+ LOC
- 400+ commits

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

## **Punch Card**

Fullfilled criteria







Workflow

Changes in design