Re-building Sakuli with node

Things we learned about native node add-ons and the tears along the way

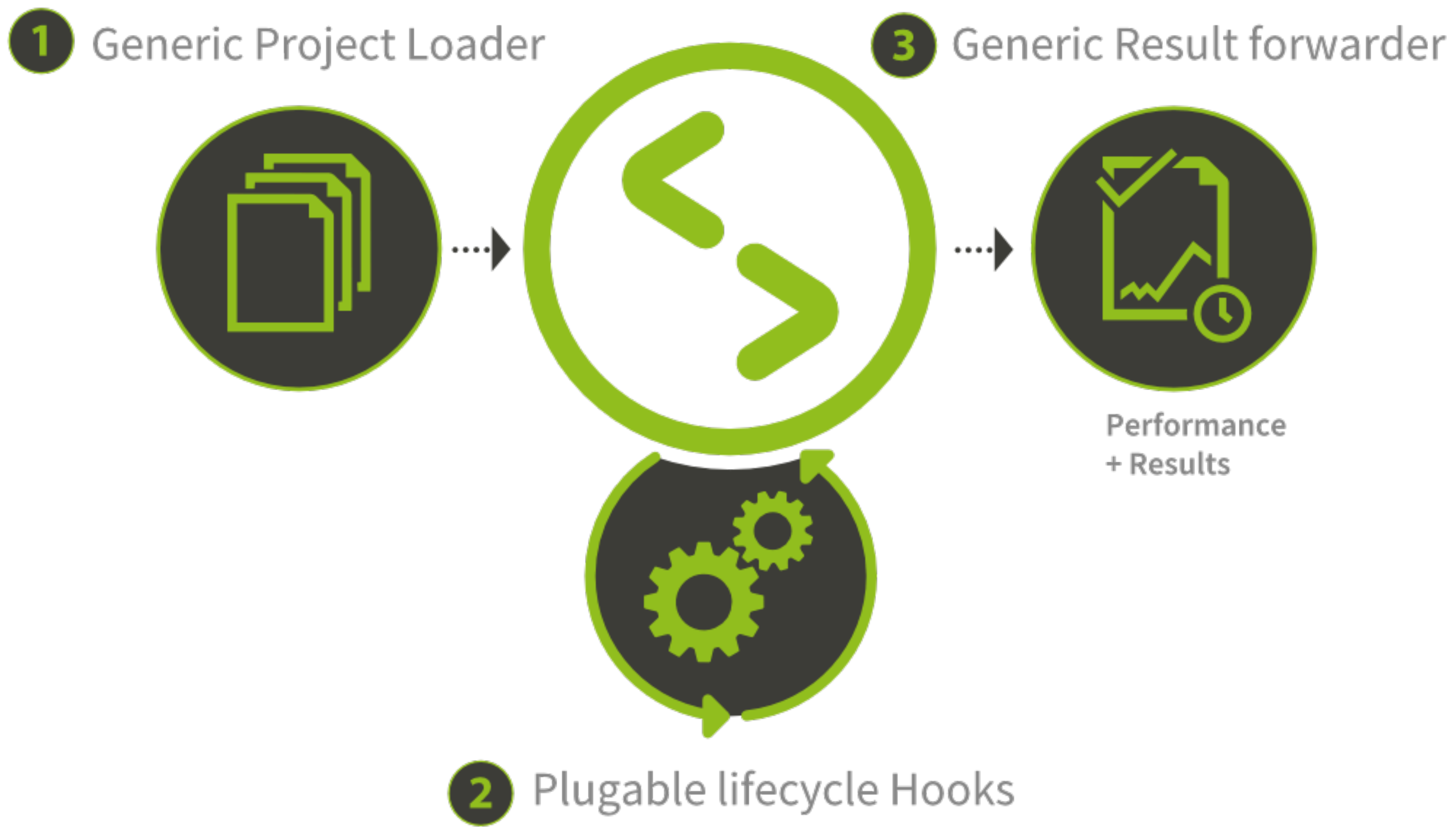
Agenda

- About
- Native Addons
- Packaging Additional Dependencies
- Even More Pitfalls

About

Sakuli

- Sakuli focuses on flexibility and extensibility
- At its core, Sakuli is a generic testrunner
- Processes combinations of
  - Project Loader
  - Context Provider
  - Result Forwarder
- Offers flexible configuration and own extensions



Sakuli

- A framework to test and / or automate browsers and desktops
- Runtime: Node.js
- Web Testing: Selenium / Webdriver
- Native Testing: nut.js
- Programming Language
    - Framework: JavaScript (TypeScript)
    - Tests: JavaScript
- Installation: npm

Native Addons

Native Addons

- Dynamically-linked shared objects
- Loaded into node using require()
- Two ways of building native addons:
  - Native Abstractions for Node.js (nan)
  - N-API
- Written in C / C++
- Built for a specific target platform

Native Addons - Build Tools

- node-gyp: https://www.npmjs.com/package/node-gyp
  - Bundles the **gyp** project used by the Chromium project
- cmake-js: https://github.com/cmake-js
  - Wrapper around **cmake**

Native Addons - nan vs. N-API

nan

- Depends on V8 APIs
- V8 APIs are not guaranteed to be stable over node releases
- Add-on has to be re-compiled for every node ABI (https://nodejs.org/en/download/releases/)

N-API

- No longer dependent on a specific JavaScript runtime (e.g. V8)
- Aims to provide ABI stability
  - Add-on compiled against a certain version will not have to be re-compiled for subsequent versions

Native Addons - nan vs. N-API

- nan
  - # of bindings = # of supported platforms * # of supported ABI versions
  - # of CI jobs = # of supported platforms * # of supported ABI versions
- N-API
  - # of bindings = # of supported platforms
  - # of CI jobs = # of supported platforms

Native Addons - Distribution

# of platforms * # of supported ABI = too much packages

Native Addons - Distribution

- Fortunately, there's tooling!
  - https://www.npmjs.com/package/node-pre-gyp
  - https://www.npmjs.com/package/prebuild
  - https://github.com/prebuild/prebuild-install

- prebuild:
  - Builds your native addon for a given platform and ABI version
  - Pushes the binary to a GitHub release
- prebuild-install:
  - Will determine your platform and ABI version
  - Download the respective binary from GitHub
  - Rebuild from source on error

Packaging Additional Dependencies

Additional Dependencies

- Have to be available during compiletime
- Have to be available during runtime
    - Runtime dynamic linker has to be able to find the lib

- So we have to:
    - Build / install our dependency
    - Link against our dependency when compiling the addon
    - Make sure the dependency is available at runtime

Additional Dependencies - Case Study

- Dependency: OpenCV
  - Step 1: npm project to build OpenCV
  - Step 2: Run several CI jobs for macOS, Linux and Windows
  - Step 3: Publish packages for each platform to npm

Additional Dependencies - Case Study

- Native addon: opencv4nodejs
  - Step 1: On install, run script to install platform dependent OpenCV libs
  - Step 2: Run CI jobs for macOS, Linux and Windows with supported node versions
  - Step 3: Push prebuilt addons to GitHub release

Additional Dependencies - Case Study

- Actual library
    - Step 1: Install native addon via npm
    - Step 2: Done!

Pitfalls

Additional Dependencies - Case Study

- Pitfall 1:
  - npm pack: https://github.com/npm/npm/issues/3310
  - npm pack follows symlinks, but does not include them
    - Leads to missing links to libs
    - Causes runtime errors due to failed library lookup

Additional Dependencies - Case Study

- Pitfall 1:
  - npm pack: https://github.com/npm/npm/issues/3310
  - npm pack follows symlinks, but does not include them
    - Leads to missing links to libs
    - Causes runtime errors due to failed library lookup
- Pitfall 2:
  - GNU ld vs. BSD ld
  - The dynamic runtime linker needs to be able to find libs
    - $origin (GNU ld) vs. @loader_path (BSD ld)

Sakuli - Links

- https://sakuli.io
- https://github.com/sakuli
- https://github.com/nut-tree/nut.js
- https://twitter.com/sakuli_e2e

Any questions?

Thank you!

consol

Wir unternehmen IT

ConSol
Consulting & Solutions Software GmbH

St.-Cajetan-Straße 43
D-81669 München
Tel.: +49-89-45841-100
info@consol.de
www.consol.de
Twitter: @consol_de