



December 11-12, 2019  
Montreal, Canada

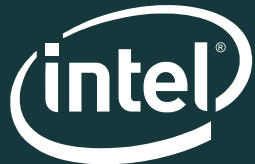
# Next Generation N-API

*A hands-on workshop*

N-API Team



Gabriel Schulhof



 @gabrielschulhof

 gabriel.schulhof@intel.com

## Works at Intel



### Involved with the API working group

- Promises
- Exception handling
- Environment propagation
- Module loading
- Wrap/Unwrap



Michael Dawson



 @mhdawson1  
 @mhdawson

## IBM Community Lead for Node.js Active Node.js community member

- Technical Steering Committee TSC member
- Community Committee member
- n-api, build, security, benchmarking, diagnostics, release, user-feedback, teams and WGs.



Jim Schlight

inspiredware

 @inspiredware  
 @jschlight

Head of a consultancy based in Ashland  
Member of the N-API Working Group

- node-pre-gyp
- Prebuild
- Documentation



Nicola Del Gobbo



 @NickNaso  
 @NickNaso

Developer at Packly  
Member of the N-API Working Group

# CONTRIBUTORS

**Anna Henningsen**  
@addaleax

**Gabriel Schulhof**  
@gabrielschulhof

**Hitesh Kanwathirtha**  
@digitalinfinity

**Jim Schlight**  
@jschlight

**Micheal Dawson**  
@mhdawson

**Nicola Del Gobbo**  
@NickNaso

**Kevin Eady**  
@KevinEady

**Arunesh Chandra**  
@aruneshchandra

**Taylor Wall**  
@boingoing

**Anisha Rohra**  
@anisha-rohra

**Kyle Farnung**  
@kfarnung





# OBJECTIVES OF THE WORKSHOP



Orientation to N-API



Awareness of available tools and processes



A good start on your own projects





# **WORKSHOP SCHEDULE**

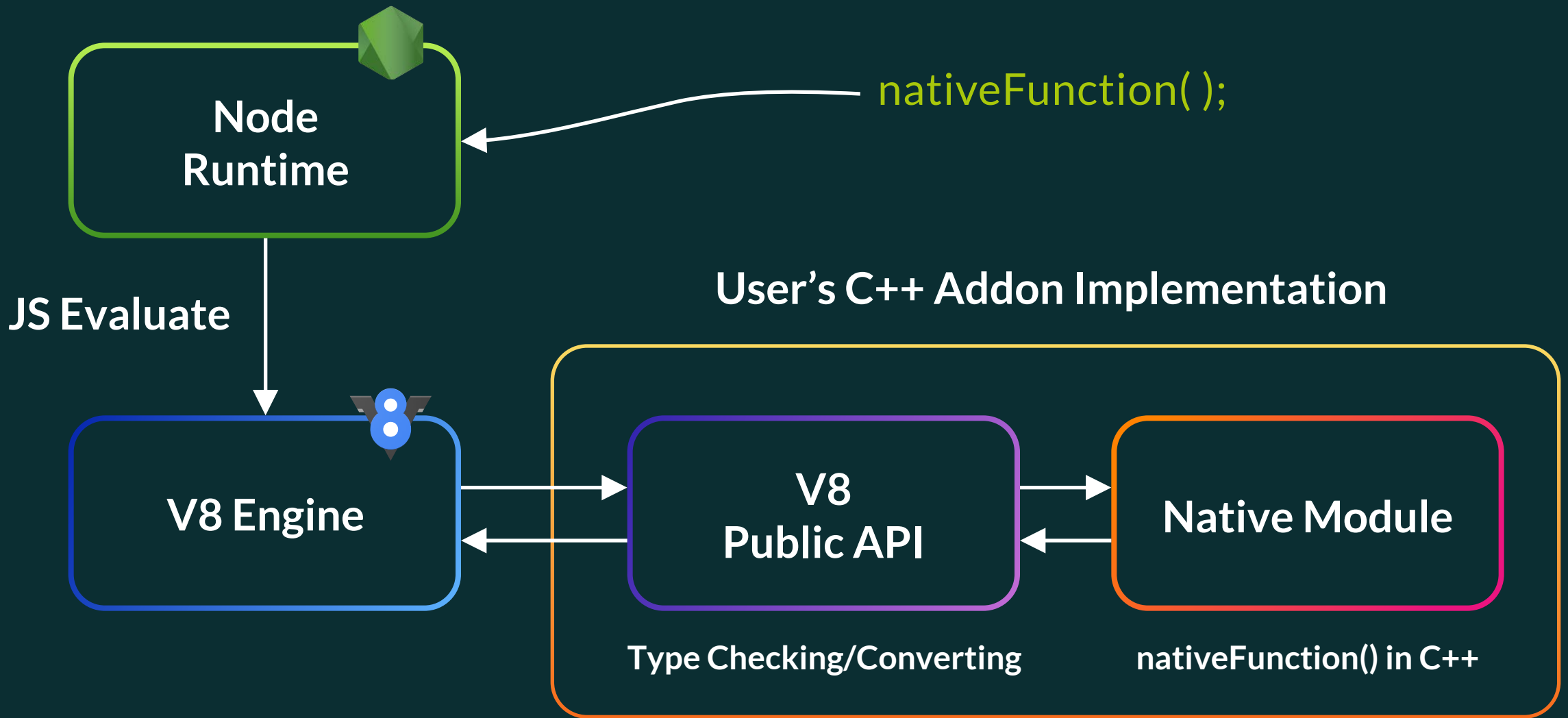
- 1. Introduction to N-API and node-addon-api**
- 2. Online tutorials**
- 3. Let's port some modules/individual projects**
- 4. Wrap-up and assessment**



**WHAT IS A  
NATIVE ADDON?**

Node.js Addons are **dynamically-linked shared objects**, written in **C++**, that can be loaded into Node.js using the **require()** function, and used just as if they were an ordinary Node.js module.

They are used primarily to provide an **interface** between **JavaScript** running in Node.js and **C/C++** libraries.



What is a native addon?

# MOTIVATIONS FOR N-API

The API to implement native add-ons has been changed across different version of Node.js

Most of the changes were on **V8 API** and **ObjectWrap API** and other node internals

About **30%** of modules depend on a native add-on. A breakage on a native addon could become very important e.g. **node-sass**

# Need an adapter to stay compatible across different versions of Node.js

## **NAN** - Native Abstraction for Node.js

- API compatibility
- Strongly bonded with **V8 API**
- You have to recompile your native add-ons when switching to a different version of Node.js



# End user

```
| was compiled against a different Node.js version using  
| NODE_MODULE_VERSION 51. This version of Node.js requires  
| NODE_MODULE_VERSION 57. Please try re-compiling or re-installing  
| the module (for instance, using `npm rebuild` or `npm install`).
```

# Maintainers



was compiled against a different Node.js version using

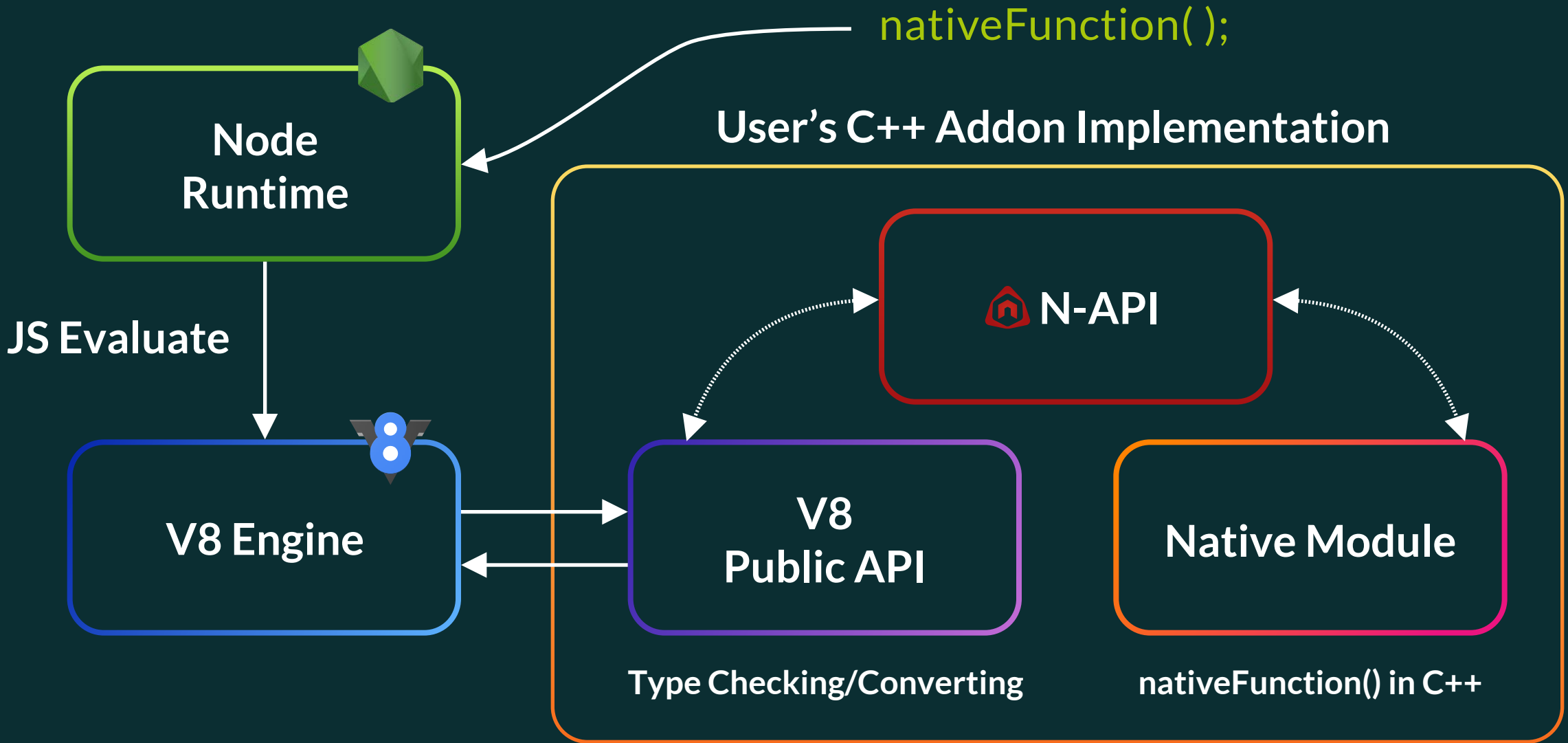
#117

was compiled against a different Node.js version using NODE\_MODULE\_VERSION 59. This version of Node.js requires NODE\_MODULE\_VERSION 57. Please try re-compiling or re-installing the module (for ...

JCMais/node-libcurl Opened by gsion on 6 Mar 1 comment

# WHAT'S N-API

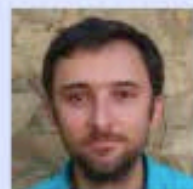
- ⦿ Abstraction of the underlying JavaScript engine
- ⦿ Defines and exports C types and functions that are independent from the JavaScript engine
- ⦿ A binary-stable ABI



# Without N-API

Addon

```
Node.js: 6.12.2  
V8: 5.1.281  
NODE_MODULE_VERSION: 48
```



# Without N-API

- ABI break
  - At worst, mysterious segfaults
  - At best, addon fails to load
  - NODE\_MODULE\_VERSION mismatch

```
Node.js: 8.9.3  
V8: 6.1.534  
NODE_MODULE_VERSION: 57
```



# With N-API

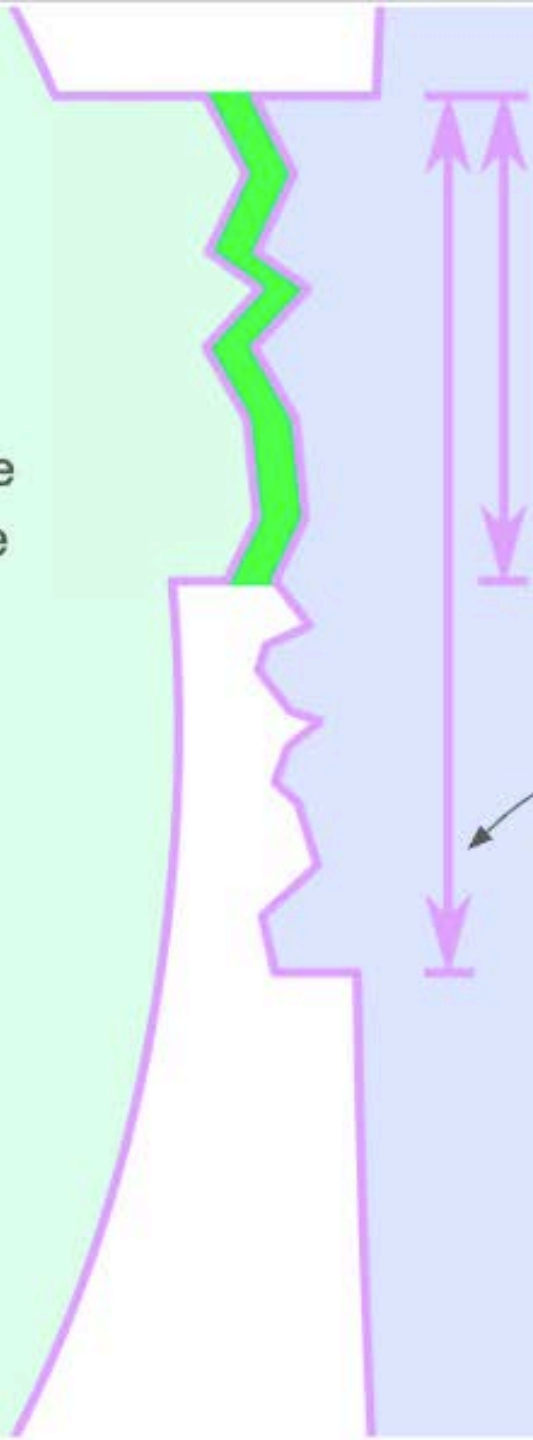


Node.js: no matter  
V8/ChakraCore: no matter  
NAPI\_VERSION: 1



# With N-API

- No ABI break
  - `NAPI_VERSION` is cumulative
  - Addon is forwards compatible



Node.js: later  
V8/ChakraCore: no matter  
NAPI VERSION: 2

