Extending Node.js Using C++

Kenneth Geisshirt
Open Source Days 2013

About me

- Education
  - B.Sc. in computer science (Univ. of Copenhagen)
  - M.Sc. in chemistry (Univ. of Copenhagen)
  - Ph.D. in soft material science (Roskilde Univ.)
- Freelance writer and tech review
- Senior software developer at TightDB, Inc.
  - Documentation and benchmarking
  - Implementing language bindings

Agenda

- What is Node.js and V8?
- C++ classes
- Wrapping classes
  - Setters, getters, deleters, enumerators
  - Anonymous functions
  - Exceptions
  - Instantiate objects
- Building extensions

$\left.\begin{array}{l}\\ \\ \\ \\ \end{array}\right\}$ Code examples

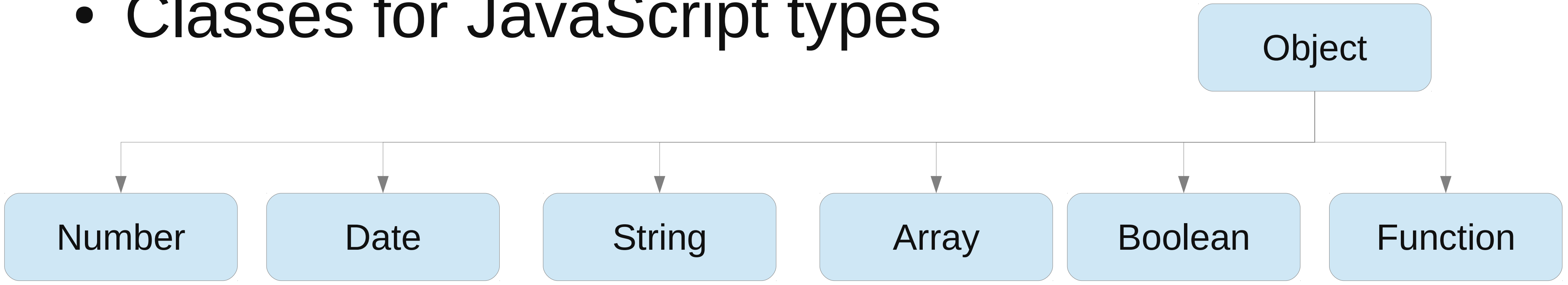
What is Node.js?

- Server-side JavaScript
- Based on Google V8
- Event-driven, non-blocking I/O
- Many modules
  - Network, files, databases, etc.
  - Mostly written in JavaScript

```
var http = require('http');
var server = http.createServer(function(req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    var q = require('url').parse(req.url, true);
    res.end('<html><body>Hello  '+q.query.name+'</body></html>');
}).listen(9876, "127.0.0.1");
```

Google V8

- High performance JavaScript engine
  - Written in C++
  - Incremental garbage collector
  - Just-in-Time compilation (ARM, IA-32, x86_64, MIPS)
  - Used in Chrome and Chromium
- Classes for JavaScript types



C++ classes

- Person
  - `firstname`
  - `lastname`
  - `birthday`
  - `to_str`
- Book
  - `add`
  - `lookup`
  - operator `[]`
  - `remove`
  - `size`

Files:

```
book.hpp, book.cpp
person.hpp, person.cpp
main.cpp
Makefile
```

Wrapper classes

- Inherit from ObjectWrap
- Declaring friendships can be an advantage
- Common (static) methods:
  - Method `Init` adds class to runtime
  - Method `New` instantiates an object
- Remember that JavaScript has “funny” scope rules
- Special exception class
- Validate arguments as JavaScript isn't strongly typed

Wrapper classes

- Inherit from ObjectWrap
- Declaring friendships can be an advantage
- Common (static) methods:
  - Method `Init` adds class to runtime
  - Method `New` instantiates an object
- Remember that JavaScript has “funny” scope rules
- Special exception class
- Validate arguments as JavaScript isn't strongly typed

Initialize

- Method `Init`
  - Sets the class name
  - Sets the number of internal fields
  - Adds methods to runtime
    - `NODE_SET_PROTOTYPE_METHOD` macro
  - Adds getter/setter/deleter/enumerator
  - Create a constructor (function object)

Example: `PersonWrap::Init` and `BookWrap::Init`

Arguments

- Class `Arguments` are in methods
  - An array of V8 objects
- Variable number of arguments
  - `Length` method is useful
- Typical a lot of input validation
  - `IsString`, `IsArray`, `IsNumber`, etc.
- The `This()` method returns the current object
  - You must unwrap it to get your object

Example: `BookWeap::Lookup`

Scope

- Methods need access to the JavaScript stack
- A `HandleScope` object can help you
  - Methods begin by creating the object
  - Stack allocation (local variable)
- Exit methods by closing scope
  - Returns a variable to the previous scope
  - Scope cannot be used
  - Garbage collector will eventually deallocate it
- `Local<T>` is for local (stack allocated) variables

Photo by Hertje Brodersen

Example: `BookWrap::Length`

(V8) Exceptions

- In JavaScript, you can throw any object
- V8 implements it as a C++ class
- Five different:
  - RangeError, ReferenceError, SyntaxError, TypeError, Error
- You throw by returning an exception object
  - And it can be caught in you JavaScript program

Example: BookWrap::Add

New instance

- JavaScript programs can create new objects
  - Or instances of you class
- The `New` method is called when a new object is created
  - Create a wrapper object
  - Probably you must create a wrapped object, too
  - Wrap `this` and return it
- The constructor can easily take arguments

Example: `BookWrap::New`

New instance from C++

- You can create JavaScript objects from C++
  - Useful when a method returns a wrapped object
- Overload the `New` method
  - Or use another name
- The constructor (of the wrapper class) has a `NewInstance` method
- Pointer to wrapped object is added as internal field
- Friendship between wrapper classes is very useful

Example: `PersonWrap::New` ×3

Getter and setter

- JavaScript has to index operators
  - `[]` for array-like access (indexed)
  - `.` for attributes (named)
- No negative index (uint32_t)

Examples: `BookWrap::Getter` and `PersonWrap::Setter`

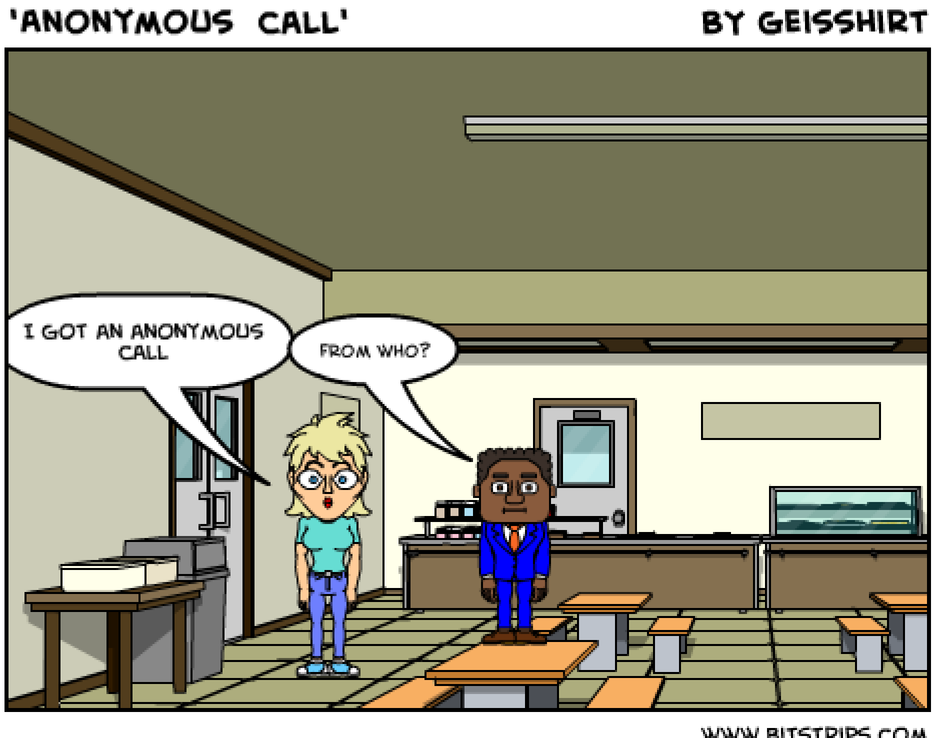
Deleter and Enumerator

- JavaScript's `delete` operator is supported by implementing a Deleter
  - Must return either true or false
- An enumerator makes it possible to do `for ... in`

Example: `BookWrap::Deleter` and `BookWrap::Enumerator`

Anonymous functions

- JavaScript programmers love anonymous functions
  - Functions are just an object → can be an argument
- You must set the context
  - JavaScript is complex
  - Current one is often fine
- Set up the arguments



Example: `BookWrap::Each`

Catching Exceptions

- JavaScript functions can throw exceptions
  - And you can catch them in C++
  - The TryCatch class implement a handler
- Complication:
  - JavaScript might return an object if successful
  - But an exception is also an object
  - (the V8 tutorial is probably wrong)
- You can rethrow exceptions

Example: `BookWrap::Apply`

Build

- Initialize classes from `init`
  - File: `funstuff_node.cpp`
- Using old-school node-waf
  - Write a `wscript` file
- Newer extensions use gyp

Where to go?

- Get my demo extension: https://github.com/kneth/FunStuff
- Node.js: http://nodejs.org/
- *JavaScript Unit Testing* by Hazam Salah
- V8 classes: http://bespin.cz/~ondras/html/hierarchy.html

Observations

- Extensions do not have to be a one-to-one mapping
- A lot of code to do input validation
  - JavaScript is not strongly typed!
- C++ has classes – JavaScript doesn't
  - Awkward for JavaScript programmers
- Node.js extension can (relative) easily be ported to Chrome
- Nodeunit is nice for unit testing