Node.js Addon

Kyoungho An
Dept. of EECS, Vanderbilt University
July 25, 2012

Presentation Roadmap

- Node.js Addon?
- wscript
- Node.js Addon Examples
  - Hello world
  - Function arguments
  - Wrapping C++ objects
- Reference

Node.js Addon?

- Addons are implemented C or C++
- Addons are dynamically linked shared objects (similar to .so or .dll)
- The following knowledge is required to implement
  - V8 JavaScript
    - C++ library
    - Used for interfacing with JavaScript
  - libuv (libev + libeio)
    - C event loop library
  - Internal Node libraries
    - Wrapping objects usable in Node (node::ObjectWrap)
- Node statically compiles dependencies into executable
  - No worry about linking to used libraries

wscript

- Waf
  - Python-based framework for configuring, compiling, and installing applications

```
srcdir = '.'
blddir = 'build'
VERSION = '0.0.1'
def set_options(opt):
    opt.tool_options('compiler_cxx')
def configure(conf):
    conf.check_tool('compiler_cxx')
    conf.check_tool('node_addon')
def build(bld):
    obj = bld.new_task_gen('cxx', 'shlib', 'node_addon')
    obj.target = 'hello'
    obj.source = 'hello.cc'
```

Hello world (hello.js)

```
var addon = require('./build/Release/hello');


console.log(addon.hello()); // 'world'
```

Hello world (hello.cc)

```
#include <node.h>
#include <v8.h>
using namespace v8;

Handle<Value> Method(const Arguments& args) {
  HandleScope scope;
  return scope.Close(String::New("world"));
}

void init(Handle<Object> target) {
  target->Set(String::NewSymbol("hello"),
      FunctionTemplate::New(Method)->GetFunction());
}
NODE_MODULE(hello, init)
```

Function arguments (function.js)

```
var addon = require('./build/Release/addon');

console.log( 'This should be eight:', addon.add(3,5) );
```

Function arguments (function.cc)

```
#include <node.h>
using namespace v8;

Handle<Value> Add(const Arguments& args) {
  HandleScope scope;

  if (args.Length() < 2) {
    ThrowException(Exception::TypeError(String::New("Wrong number of
arguments")));
    return scope.Close(Undefined());
  }

  if (!args[0]->IsNumber() || !args[1]->IsNumber()) {
    ThrowException(Exception::TypeError(String::New("Wrong arguments")));
    return scope.Close(Undefined());
  }
```

Function arguments (function.cc)

```
  Local<Number> num = Number::New(args[0]->NumberValue() +
      args[1]->NumberValue());
  return scope.Close(num);
}

void Init(Handle<Object> target) {
  target->Set(String::NewSymbol("add"),
      FunctionTemplate::New(Add)->GetFunction());
}

NODE_MODULE(addon, Init)
```

Callbacks (callback.js)

```
var addon = require('./build/Release/addon');

addon.runCallback(function(msg){
  console.log(msg); // 'hello world'
});
```

Callbacks (callback.cc)

```
#include <node.h>

using namespace v8;

Handle<Value> RunCallback(const Arguments& args) {
  HandleScope scope;

  Local<Function> cb = Local<Function>::Cast(args[0]);
  const unsigned argc = 1;
  Local<Value> argv[argc] = { Local<Value>::New(String::New("hello world")) };
  cb->Call(Context::GetCurrent()->Global(), argc, argv);

  return scope.Close(Undefined());
}
```

Callbacks (callback.cc)

```
void Init(Handle<Object> target) {
  target->Set(String::NewSymbol("runCallback"),
      FunctionTemplate::New(RunCallback)->GetFunction());
}

NODE_MODULE(addon, Init)
```

Wrapping C++ objects (wrapping.js)

```
var addon = require('./build/Release/addon');

var obj = new addon.MyObject(10);
console.log( obj.plusOne() ); // 11
console.log( obj.plusOne() ); // 12
console.log( obj.plusOne() ); // 13
```

Wrapping C++ objects (wrapping.cc)

```
#include <node.h>
#include "myobject.h"

using namespace v8;

void InitAll(Handle<Object> target) {
  MyObject::Init(target);
}

NODE_MODULE(addon, InitAll)
```

Wrapping C++ objects (myobject.h)

```
#include <node.h>

class MyObject : public node::ObjectWrap {
 public:
  static void Init(v8::Handle<v8::Object> target);

 private:
  MyObject();
  ~MyObject();

  static v8::Handle<v8::Value> New(const v8::Arguments& args);
  static v8::Handle<v8::Value> PlusOne(const v8::Arguments& args);
  double counter_;
};
```

Wrapping C++ objects (myobject.cc)

```
#include <node.h>
#include "myobject.h"
using namespace v8;

MyObject::MyObject() {};
MyObject::~MyObject() {};

void MyObject::Init(Handle<Object> target) {
  // Prepare constructor template
  Local<FunctionTemplate> tpl = FunctionTemplate::New(New);
  tpl->SetClassName(String::NewSymbol("MyObject"));
  tpl->InstanceTemplate()->SetInternalFieldCount(1);
  // Prototype
  tpl->PrototypeTemplate()->Set(String::NewSymbol("plusOne"),
      FunctionTemplate::New(PlusOne)->GetFunction());
```

Wrapping C++ objects (myobject.cc)

```
  Persistent<Function> constructor = Persistent<Function>::New(tpl->GetFunction());
  target->Set(String::NewSymbol("MyObject"), constructor);
}

Handle<Value> MyObject::New(const Arguments& args) {
  HandleScope scope;

  MyObject* obj = new MyObject();
  obj->counter_ = args[0]->IsUndefined() ? 0 : args[0]->NumberValue();
  obj->Wrap(args.This());

  return args.This();
}
```

Wrapping C++ objects (myobject.cc)

```
Handle<Value> MyObject::PlusOne(const Arguments& args) {
  HandleScope scope;

  MyObject* obj = ObjectWrap::Unwrap<MyObject>(args.This());
  obj->counter_ += 1;

  return scope.Close(Number::New(obj->counter_));
}
```

Questions?

Reference

- http://nodejs.org/docs/latest/api/addons.html#addons_addons
- https://developers.google.com/v8/embed
- http://izs.me/v8-docs/main.html