NodeJS Dublin

Driving a Sphero with Node.js

Richard Rodger
@rjrodger
richardrodger.com
richard.rodger@nearform.com

Sphero Robot

- Track a robot from your web browser
- Drive it from your web browser too
- Using the Node.js JavaScript engine
- gosphero.com

nearForm

Controlling a Sphero

- Fully documented BlueTooth protocol

Device ID 02h – Sphero

These commands are specific to the features that Sphero offers.

Set Heading – 01h

	DID	CID	SEQ	DLEN	HEADING
Command:	02h	01h	<any>	03h	16-bit value

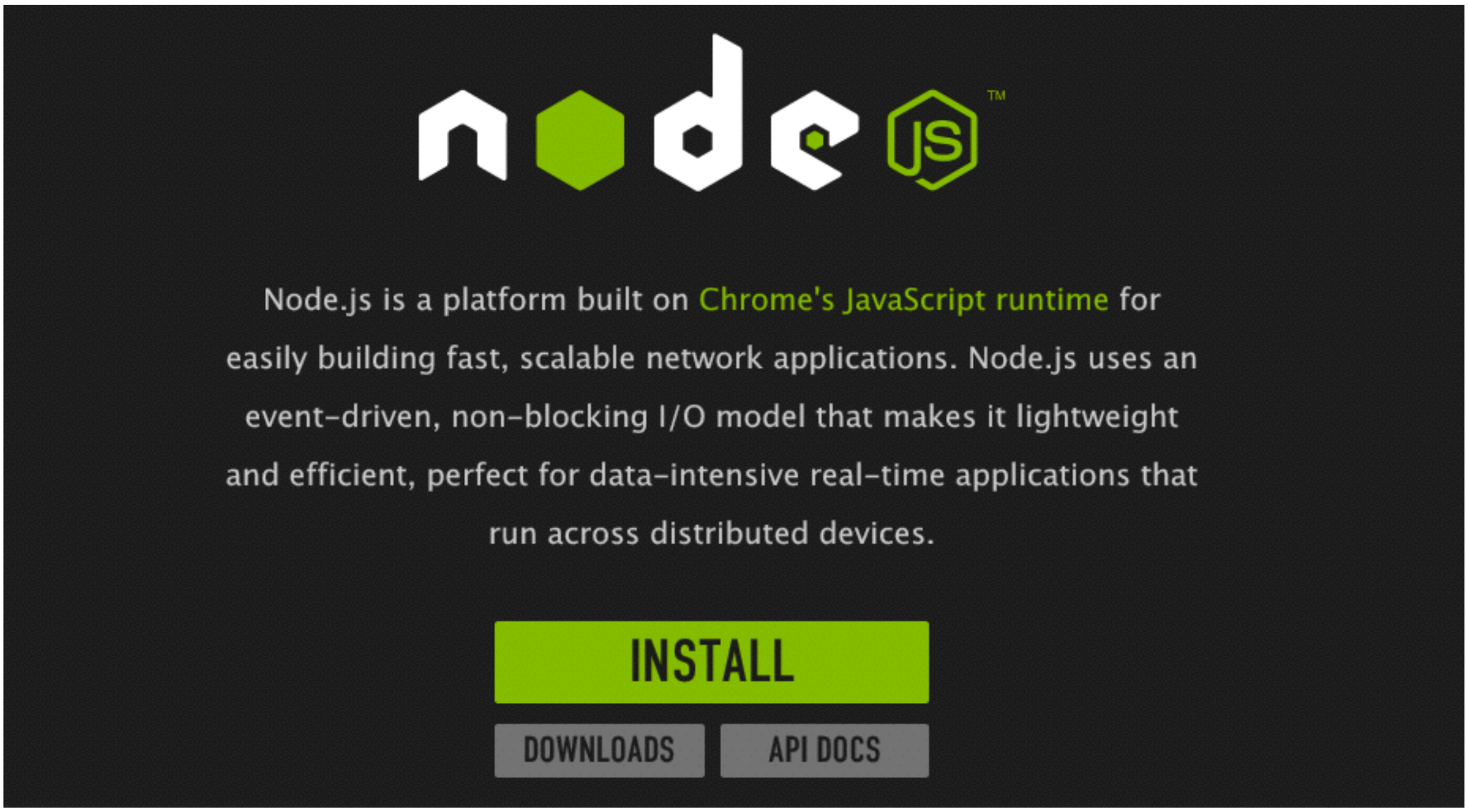
Response: **Simple Response**

This allows the smartphone client to adjust the orientation of Sphero by commanding a new reference heading in degrees, which ranges from 0 to 359. You will see the ball respond immediately to this command if stabilization is enabled.

In FW version 3.10 and later this also clears the maximum value counters for the rate gyro, effectively re-enabling the generation of an async message alerting the client to this event.

Node.js

- nodejs.org



nearForm

How Node.js Works

- Runs high-performance server-side JavaScript
- Uses the Google Chrome V8 engine
  - just-in-time compilation to machine code
  - generation garbage collection (like the Java JVM)
  - creates virtual "classes" to optimize property lookups
- Provides a minimal system level API for networking, file system, event handling, streaming data, and HTTP/S.
- Has a well-designed module system for third party code - very effective and simple to use
- Your code runs in a single non-blocking JavaScript thread
- That's OK, most of the time you're waiting for the database or network anyway!

A Simple Node.js Web Server

- A simple web server that always responds with "Hello World"

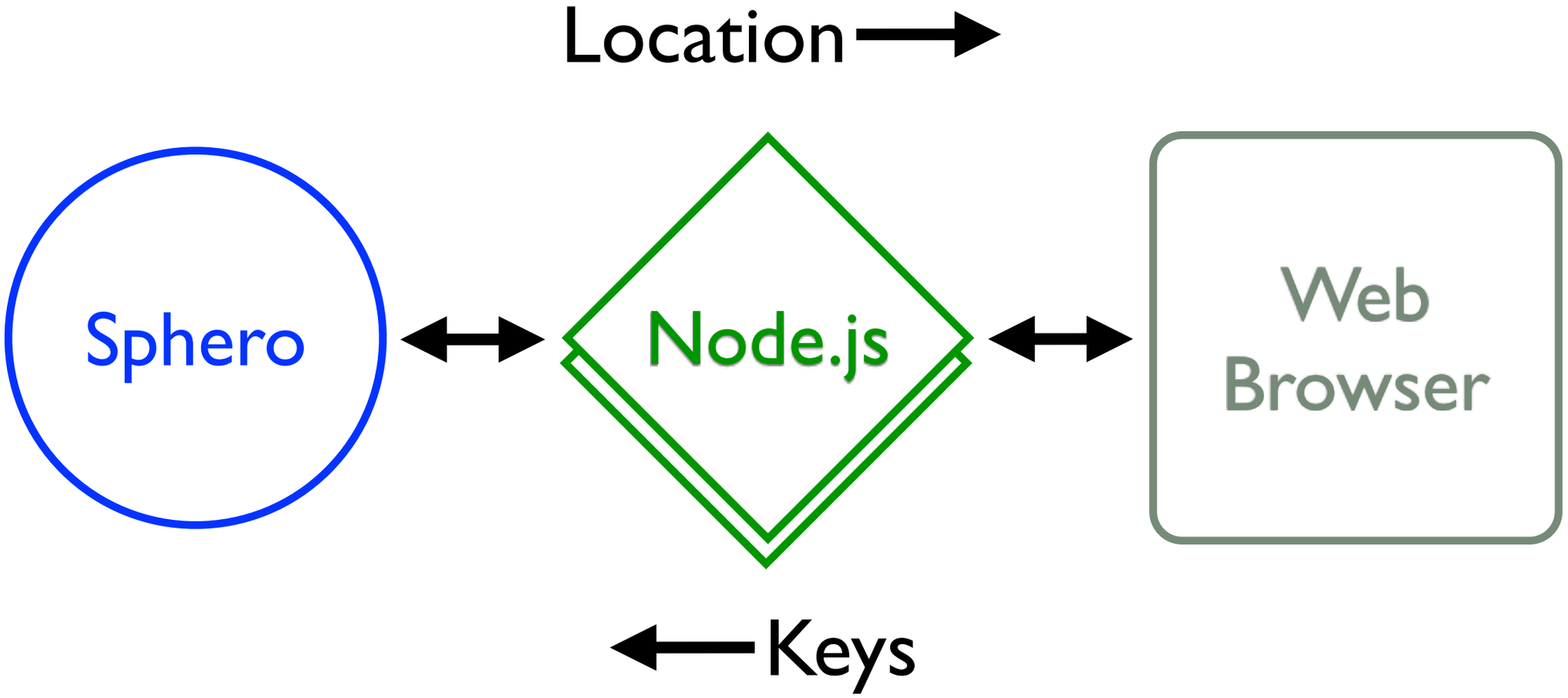
```
var http = require('http');

var server = http.createServer(function(req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
})

server.listen(1337, "127.0.0.1");

console.log('Server running at http://127.0.0.1:1337');
```

```
$ node server0.js
```

```
$ ab -c 100 -n 10000 http://127.0.0.1:1337/
```

Driving a Sphero

- The code
  - https://github.com/rjrodger/nodejsdublin-jun-2014



nearForm

The Basics

- Streams are *the way* to handle flows of data
- Built on Node's event-driven IO model
- Streams are *just like* UNIX pipes
  - That's where the idea came from!
- Provide you with a programming model
  - that is *easy to reason* about

Run: pipe0.js

- read from a file
  - create a readable stream on the file
  - use `fs.createReadStream`
- print the contents to Standard Output
  - `process.stdout` is a writable stream already

```
$ node pipe0.js
```

Run: pipe1.js

- Transforms file contents to upper case
- You can connect a series of streams
  - using `Readable.pipe(Writable)`
  - this is how data will flow through the system

```
$ node pipe1.js
```

The Core Streams

- **Readable**
  - get data (in chunks) from an input source
- **Writable**
  - send data (in chunks) to an output sink
- **Duplex**
  - both Readable and Writable
- **Transform**
  - change the data chunks (and it's Duplex)

Run: echo0.js

- Duplex streams can be piped back into themselves!
- This is how you handle two-way communication
- The idiom is:

```
duplex
  .pipe( duplex )
```

```
$ node echo0.js
```

nearForm

web0.js, front0.js, index0.html

- we need a full web server
  - **express** will do... expressjs.com
  - and let's use **browserify**...
    - your Node.js code works client-side!
      - browserify.org
- Open http://localhost:3000/index0.html

```
$ browserify public/js/front0.js > public/js/bundle0.js
$ node web0.js
```

web1.js, front1.js, index1.html

- we'll need web sockets
  - this is "real-time"
  - so let's use web sockets:
    - github.com/einaros/ws
  - we also need a streaming API
    - let's use **websocket-stream**...
      - a streaming interface for web sockets
      - github.com/maxogden/websocket-stream
- Open http://localhost:3000/index1.html

```
$ browserify public/js/front1.js > public/js/bundle1.js
$ node web1.js
```

key0.js

- we need to read key input to drive the robot
- key events are *pushed*, not *pulled*
  - use the **push** method to handle this

```
$ node key0.js
```

```
$ killall node
```

key1.js

- how do we parse key events?
  - use the **keypress** module
  - github.com/TooTallNate/keypress

```
$ node key1.js
```

```
$ killall node
```

key2.js

- Use a Transform stream to abstract key events
  - We need to define our own custom "commands"

```
$ node key2.js
```

control0.js

- Use another Transform stream to
  - react to commands
  - maintain some state
    - example: heading

```
$ node control0.js
```

control1.js

- We'll use bluetooth to talk to the real robot
  - but that's really slow for development work
  - let's generate random movement events
    - a Duplex stream can do this without getting in the way

```
$ node control1.js
```

control2.js, web2.js

- Connect it all together
  - use a TCP socket to push location data to the web server
  - and a web socket to push it to the browser
- Open http://localhost:3000/index2.html

```
$ browserify public/js/front2.js > public/js/bundle2.js
$ node web2.js
```

```
$ node control2.js
```

nearForm

control3.js, web3.js

- Show movement graphically
  - use FabricJS - a HTML canvas library
  - fabricjs.com
- Open http://localhost:3000/index3.html

```
$ browserify public/js/front3.js > public/js/bundle3.js
$ node web3.js
```

```
$ node control3.js
```

control4.js, web4.js

- Drive commands from the browser too
  - make keypresser Duplex to handle this
  - send keys back over web socket
    - github.com/madrobby/keymaster
  - more like *keymaster* and *fabricjs* here:
    - microjs.com
- Open http://localhost:3000/index4.html

```
$ browserify public/js/front4.js > public/js/bundle4.js
$ node web4.js
```

```
$ node control4.js
```

nearForm

sphero.js

- Connect the robot
  - https://github.com/rjrodger/node-sphero
  - Bluetooth pair first!
  - Use Duplex streams to capture location events, and to send commands to robot

```
$ node web4.js
```

```
$ node sphero.js
```

nearForm

Thank You!

- Learn about Node.js
  - http://blog.modulus.io/absolute-beginners-guide-to-nodejs
- Learn about Node.js streams:
  - nodeschool.io/#stream-adventure
- The code
  - https://github.com/rjrodger/nodejsdublin-jun-2014

nearForm