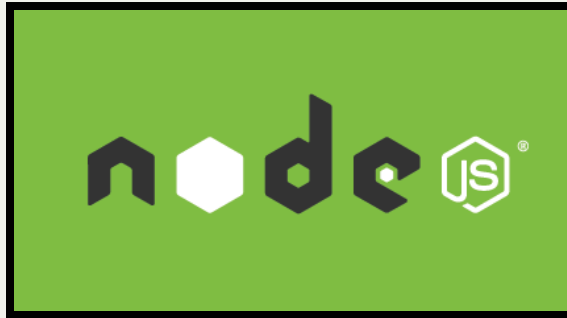
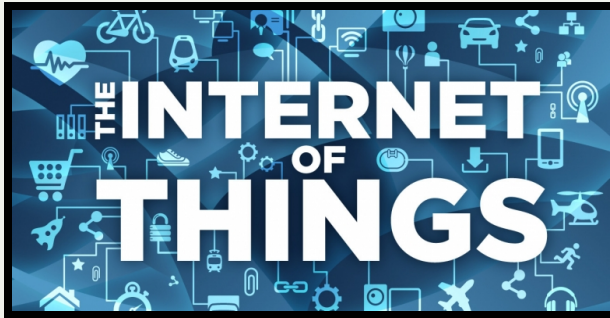


Introduction to IoT.js & JerryScript

Chunseok Lee

@ Samsung Electronics

IoT.js & JerryScript



IoT.js : Node.js-compatible platform for **IoT devices**

JerryScript : JavaScript engine for IoT devices

Dev History

2014 Mar ~ JerryScript

2015 Jan ~ IoT.js

2015 Jun ~ Open on Github

Dev Team

Samsung + Szeged Univ.

KOR + RUS + UK + GER + HUN

Please Fork Me on Github

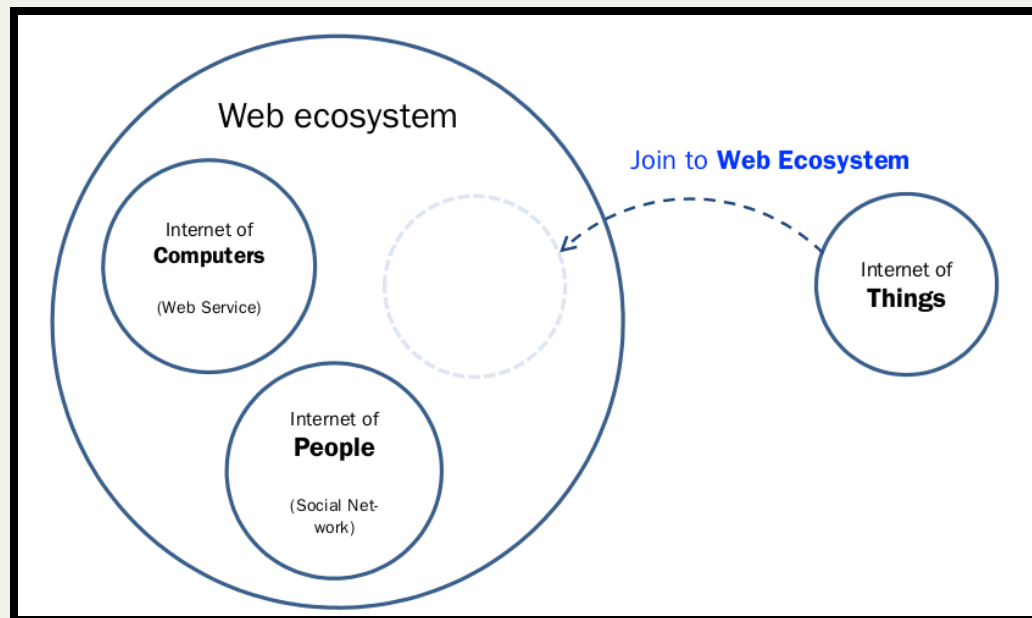
<https://github.com/Samsung/iotjs>

<https://github.com/Samsung/jerryscript>

Why JavaScript/Node.js
in constrained devices ?

Web

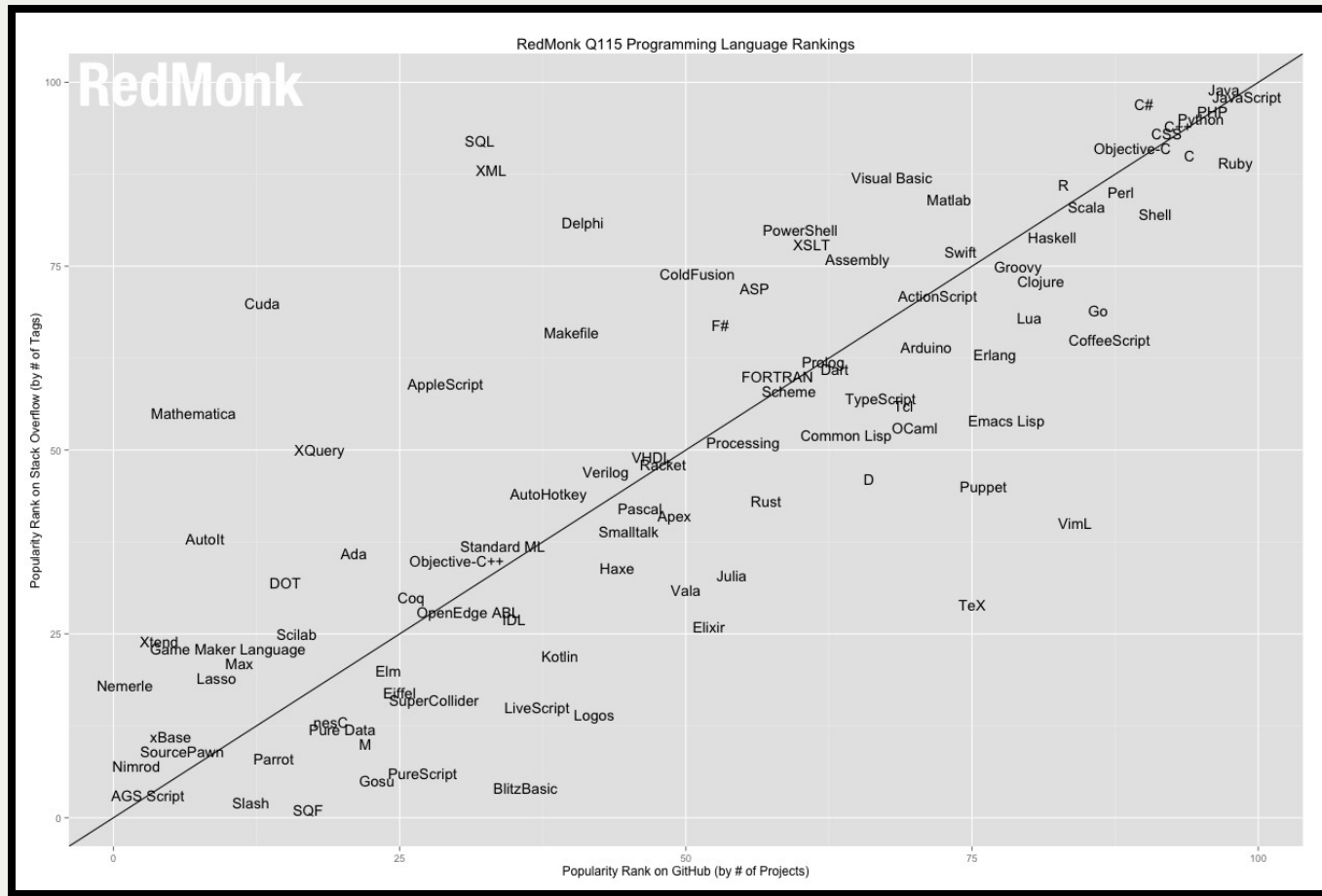
Most Successful Ecosystem in history



JavaScript

Most Popular Language

- 1 JavaScript
 - 2 Java
 - 3 PHP
 - 4 Python
 - 5 C#
 - 5 C++
 - 5 Ruby
 - 8 CSS
 - 9 C
 - 10 Objective-C
 - 11 Perl
 - 11 Shell
 - 13 R
- ...

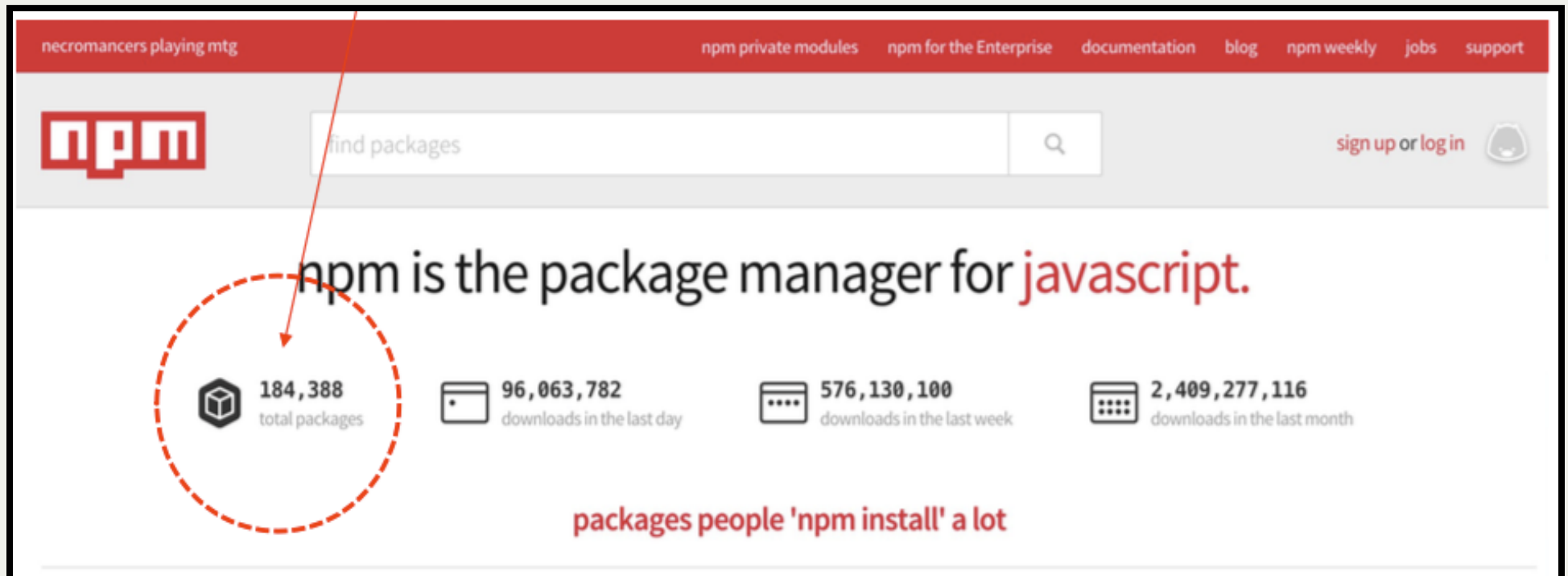


from <https://redmonk.com/sograde/2015/01/14/language-rankings-1-15/>

Node.js

Most Popular JS Platform

1. Well-known Programming Model
2. Strong ecosystem and package management system



Q. V8/Node.js on IoT devices?

IoT Hardware	Memory spec	IoT Hardware	Memory spec
 PanStamps	-RAM : 2KB -ROM : 33KB (Flash:32KB, EEPROM:1KB)	 Nanode	-RAM : 32KB -ROM
 TinyDuino	-RAM : 2KB -ROM : 33KB (Flash:32KB, EEPROM:1KB)	 Arduino Yun	-RAM : 2.5KB -ROM : 33KB (Flash:32KB, EEPROM:1KB)
 Arduino Uno	-RAM : 2KB -ROM : 33KB (Flash:32KB, EEPROM:1KB)	 mbed – LPC1768	-RAM : 32KB -ROM : 512KB
 RFduino	-RAM -ROM	 Wi-Go Module	-RAM -ROM : 2MB
 XinoRF	-RAM : 2KB -ROM : 33KB (Flash:32KB, EEPROM:1KB)	 pcDuino	-RAM : 1GB -ROM : 2GB
 OpenKontrol Gateway	-RAM : 32KB -ROM	 OpenPicus Flyport WIFI	-RAM -ROM : 16MB
 Pinoccio	-RAM : 32KB -ROM :	 Hackberry	-RAM : 512MB/1GB -ROM : 4GB
 Raspberry Pi	-RAM : 512MB -ROM	 UD00	-RAM : 1GB -ROM

Node.js 11MB flash 2MB RAM

V8 5MB flash 2MB RAM

JerryScript

JavaScript engine for IoT devices

Design Principle: Low Memory

Pure Interpreter(no JIT)

Compact Object Representation

no AST construction during parsing

Status

- Embedding API (in progress, partially supported)
- ECMAScript 5.1 Compliant : Test262 pass rate 98.5%
- Startup heap size : 2KB(x64 linux)
- bytecode snapshot support
- Low dependency on platform : own libc library
- Linux, NuttX(RTOS), OSX

Target Segment

Segment	Memory Footprint	Binary Size	Configuration	
Pico	8KB~16KB	~100KB	Compact JavaScript + Subset Profile	← ~2014
Nano	16KB~64KB	~200KB	JavaScript + Full Profile (ES v5.1)	← ~2015
Micro	64KB~256KB		JavaScript + Bytecode Optimization	
Light	256KB~	~300KB	JavaScript with JIT	← ~2016
Full	8MB~	10MB	V8	

Embedding API(1)

Execute JavaScript from your application

[illegible]

Embedding API(2)

Interaction with JavaScript environment

```
#include "jerry.h"

int
main (int argc, char * argv[]) {
    char str [] = "Hello, World!";
    char var_name [] = "s";
    char script [] = "print (s);";

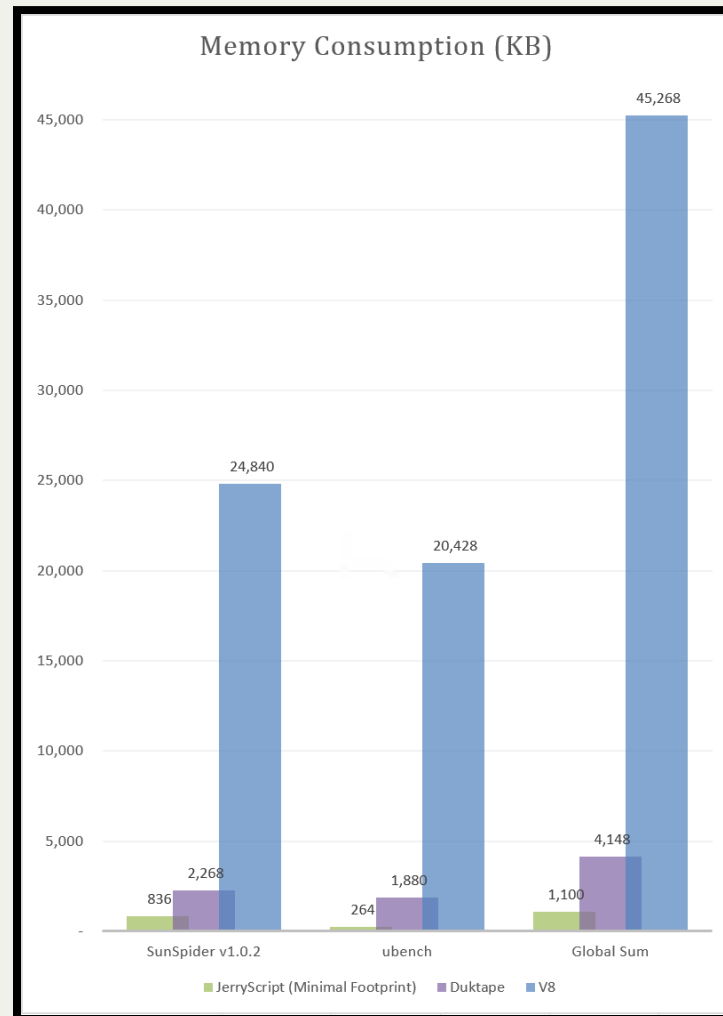
    // Initializing JavaScript environment
    jerry_init (JERRY_FLAG_EMPTY);

    // Getting pointer to the Global object
    jerry_api_object_t *obj_p = jerry_api_get_global_object ();

    // Constructing string
    jerry_api_string_t *str_val_p = jerry_api_create_string (str);
```

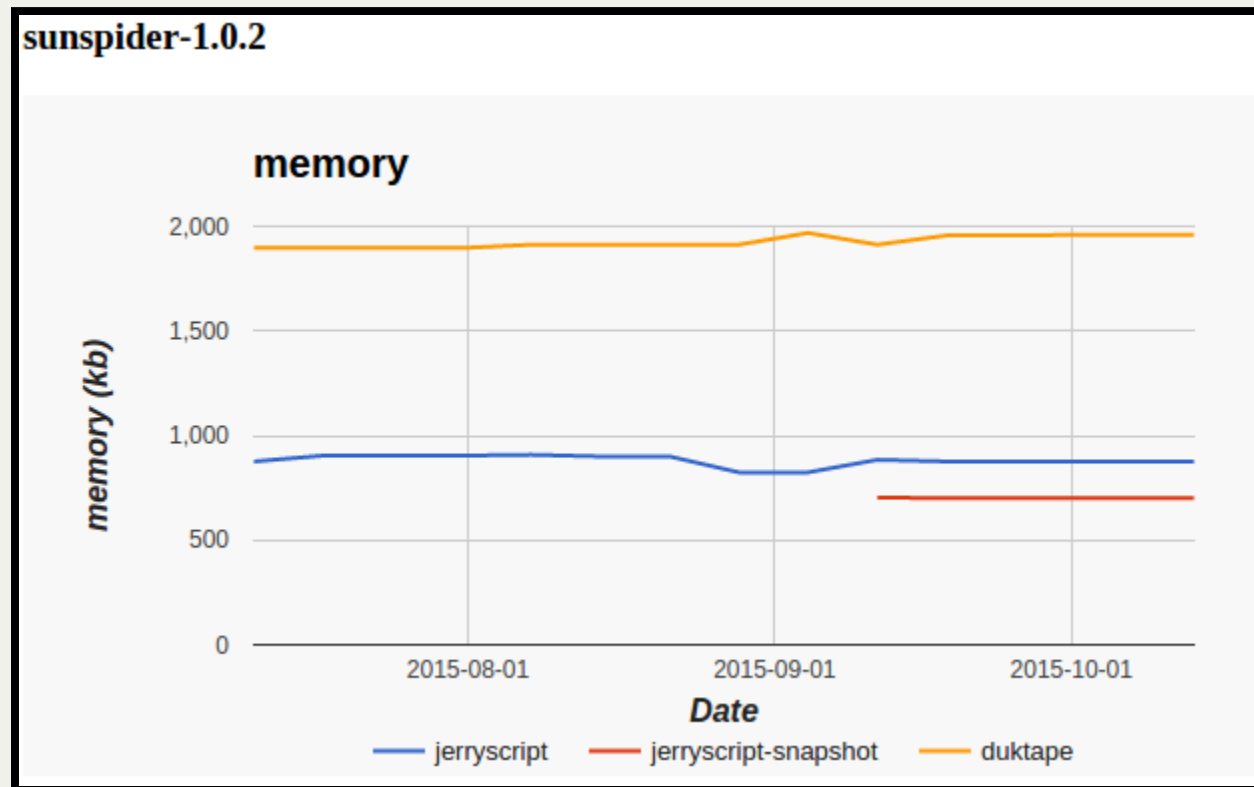
Small Memory Footprint

JerryScript/Duktape/V8

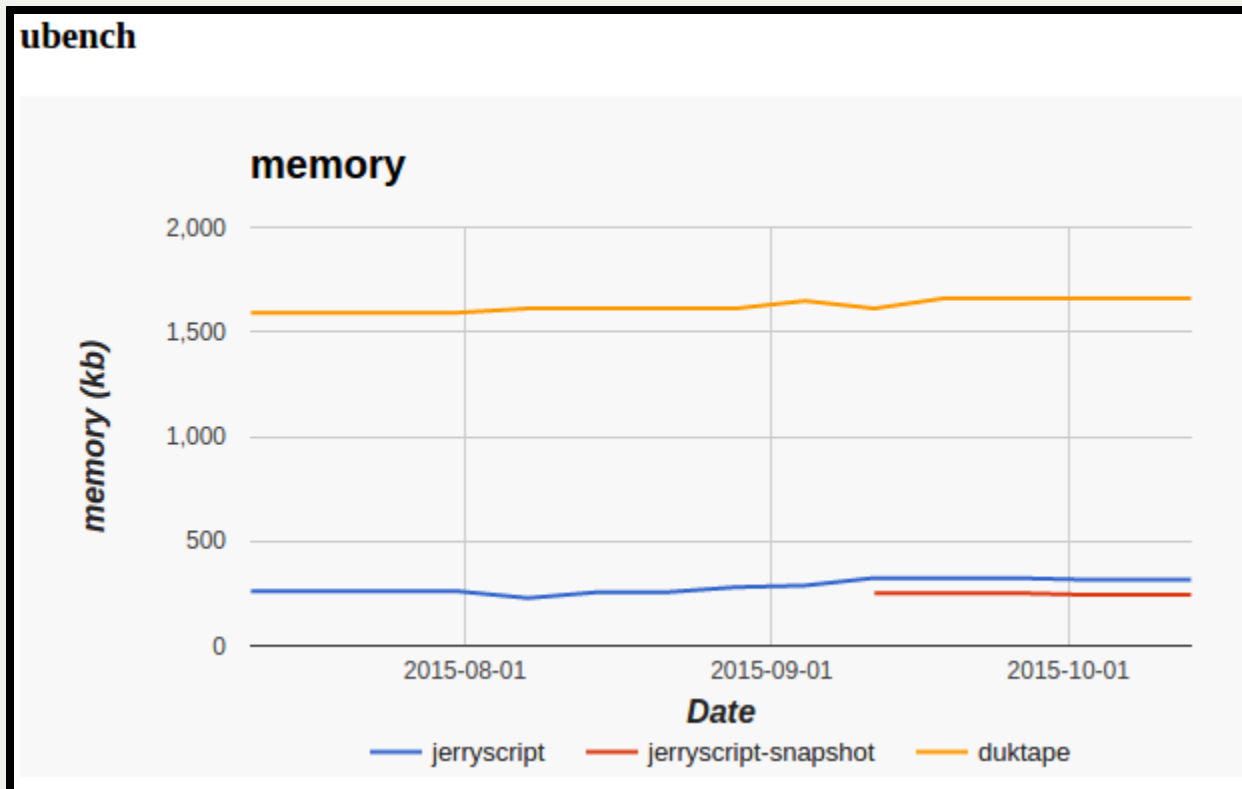


*duktape : "embeddable Javascript engine, with a focus on portability and compact footprint"

Benchmark: SunSpider



Benchmark: ubench



IoT.js

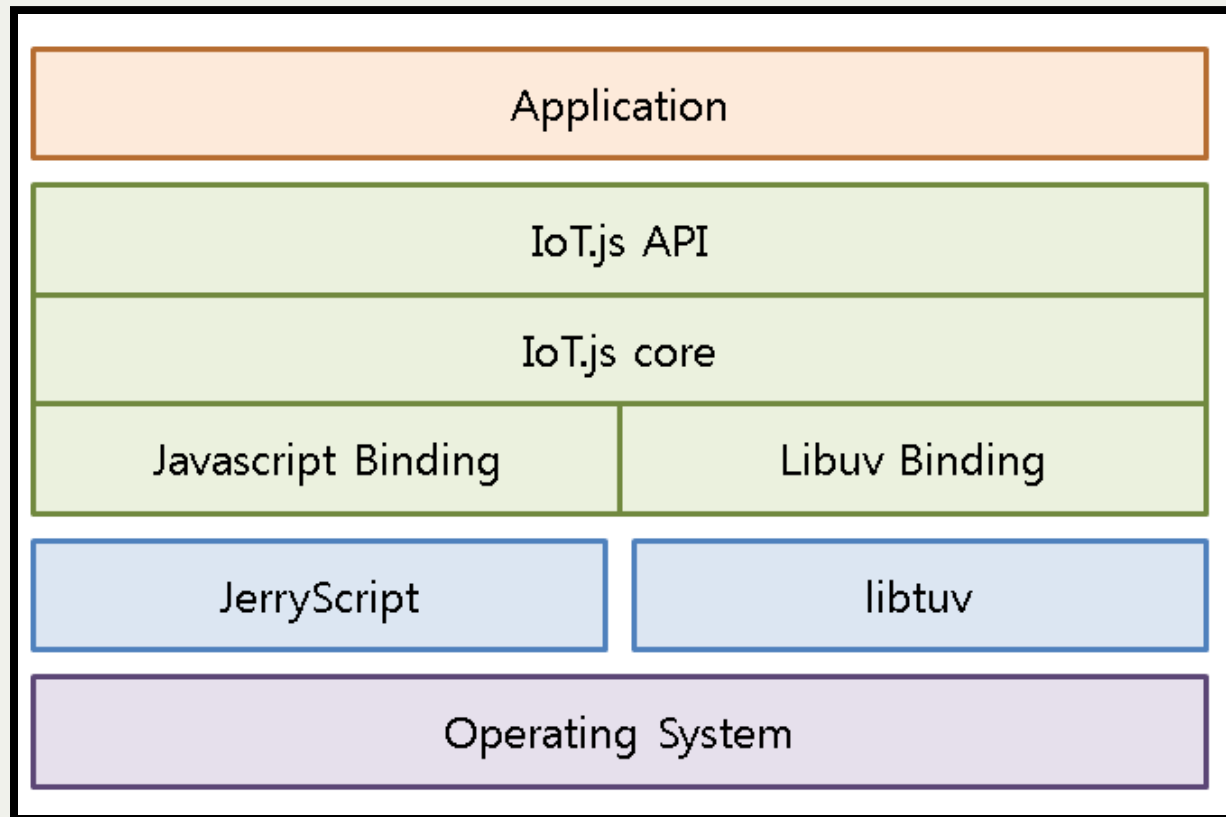
Node.js-compatible platform for IoT devices

Goal of IoT.js

Small memory consumption

Backward compatibility to Node.js app : API, (package)

IoT.js Architecture



Status: API

- Assert
- Buffer
- DNS
- Events
- File System
- HTTP
- Modules
- Net
- Process
- Stream
- Timers
- GPIO

Snapshot on IoT.js

JS heap available on STM32F4 : 78KB

Running http server on IoT.js requires 128KB JS heap

Sol: AOT bytecode construction for iot.js JS modules

Status:

memory consumption

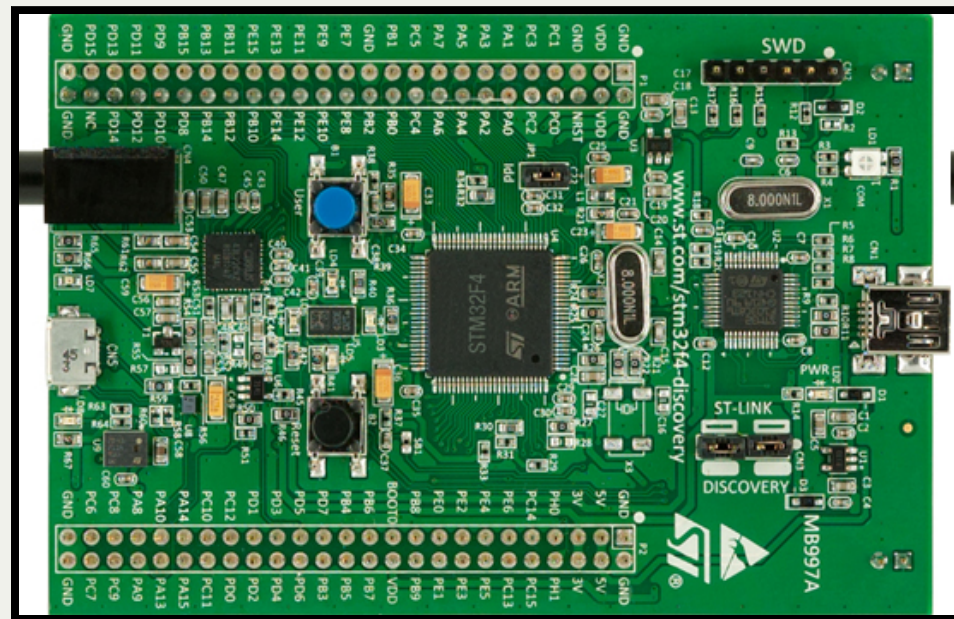
JS heap consumption

	IoT.js	Node.js
require(http)	49k bytes	2.1M
require(http+gpio)	54k bytes	-
http server + require(gpio)	71k bytes	2.4M bytes
http server with 1.7k response	76k bytes	3.1M bytes
JS heap size	77k(STM32F4)	-

Reference Hardware

STM32F4 discovery board

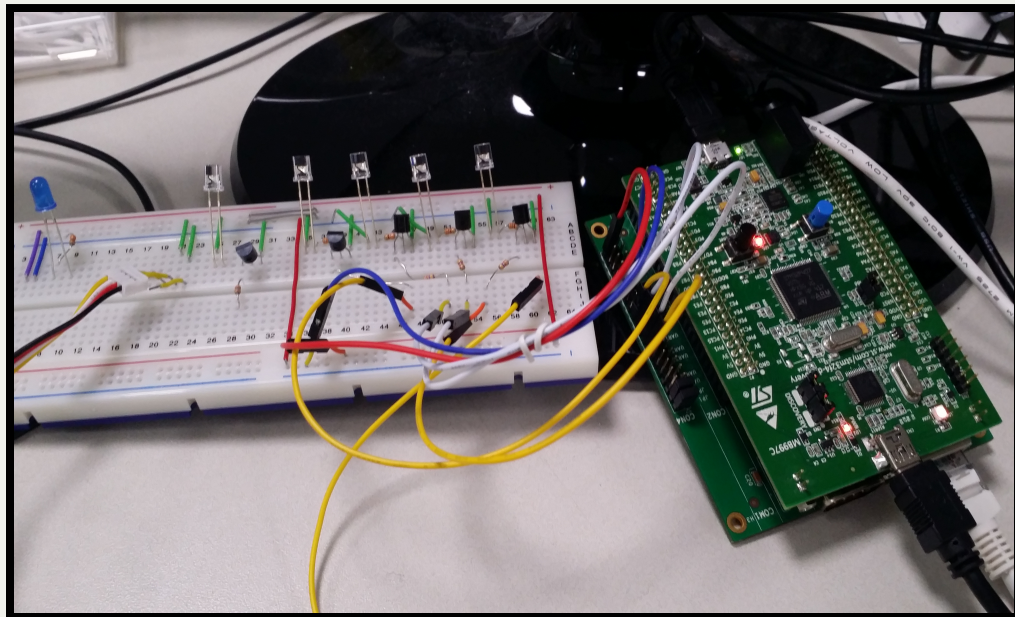
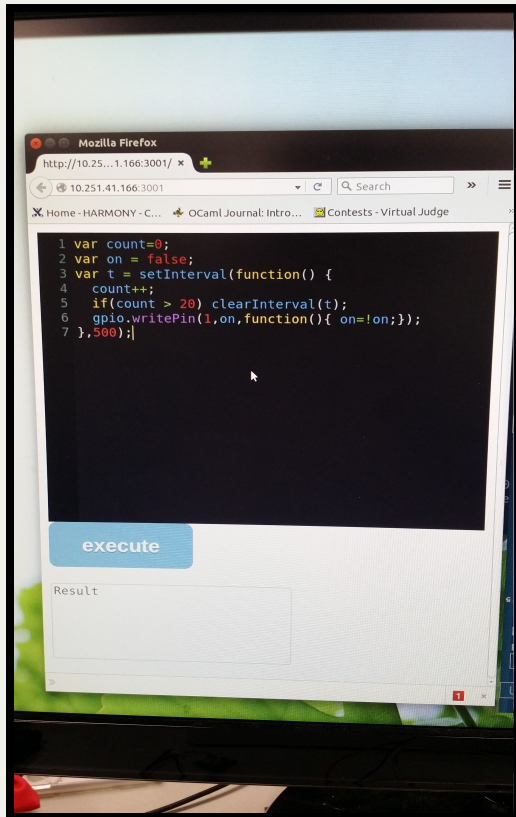
Cortex-M4F core, 128kb RAM+64kb CCM, 1M flash



Demo

JS execution via Web Browser

http server is running on STM32F4



Demo(code)

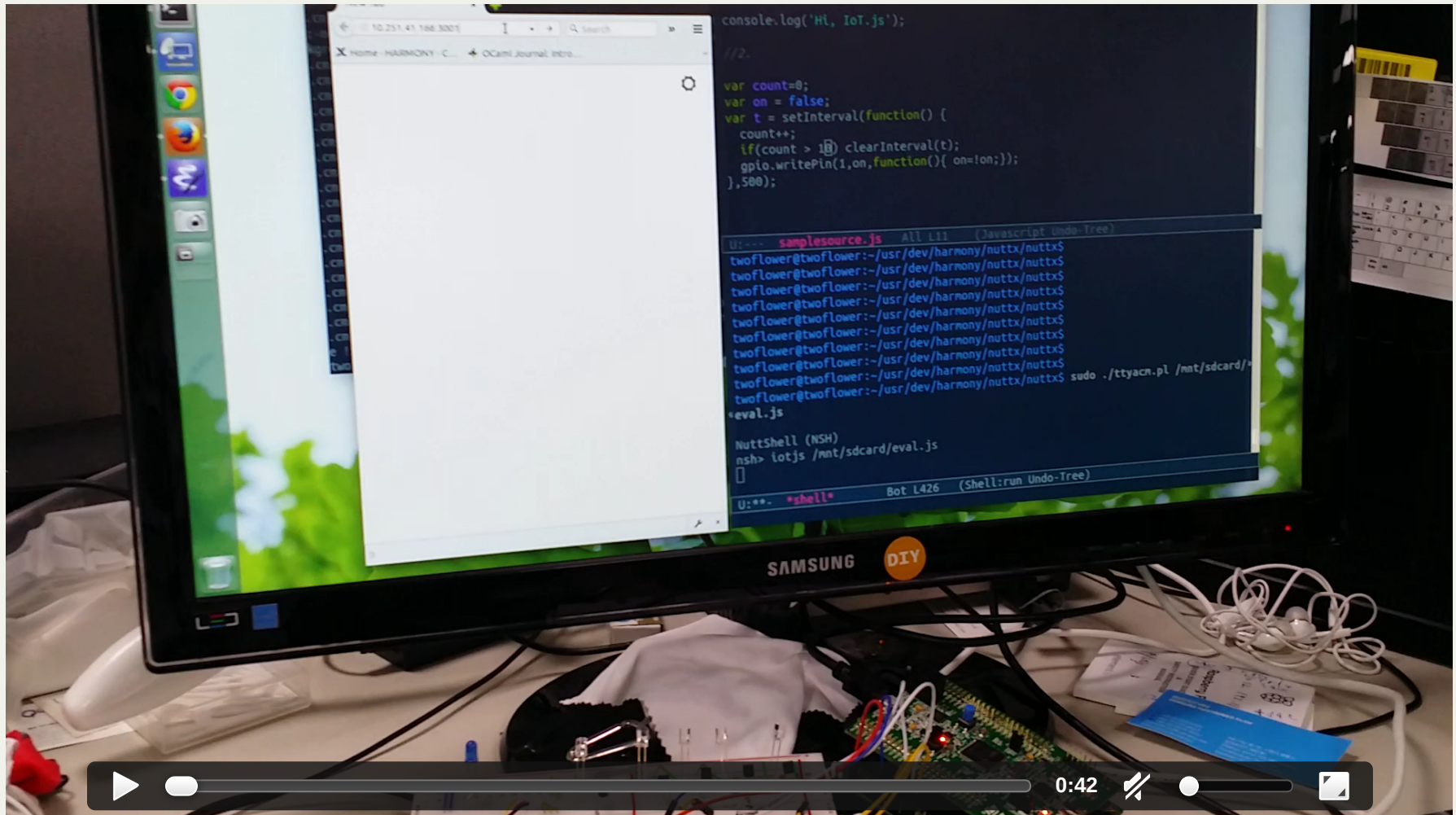
server JS code

```
var http = require('http');
var server = http.createServer(function(req, res) {
  var data = '';
  if (req.url == '/') {
    res.writeHead(200, { 'Connection' : 'close',
                        'Content-Length' : htmldata.length
                      });
    res.end(htmldata);
  }

  if (req.method == 'POST') {
    req.on('data', function(chunk) {
      console.log('data:'+chunk);
      data = data + chunk;
    });
    req.on('end', function() {
      var result = eval(data);
```

Demo(Video)

JS execution via Web Browser



Future Work

Smaller memory footprint/binary size

JerryScript Embedding API Enhancement

IoT.js package management system

Support more devices

By Your Participation

Please Fork Me on Github

<https://github.com/Samsung/iotjs>

<https://github.com/Samsung/jerryscript>

Thank You