Day 3-2

Assignment:

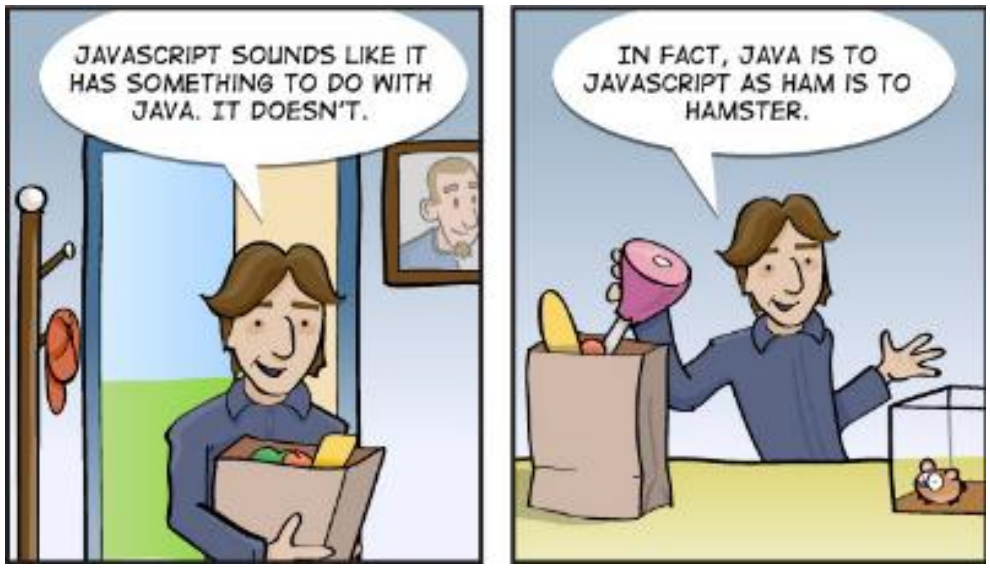
- Homework 03, Due Tuesday
-

Today's Topics:

- Crash course in JavaScript
- BoneScript
- Blinking an LED
  - js
  - The hard way

JavaScript

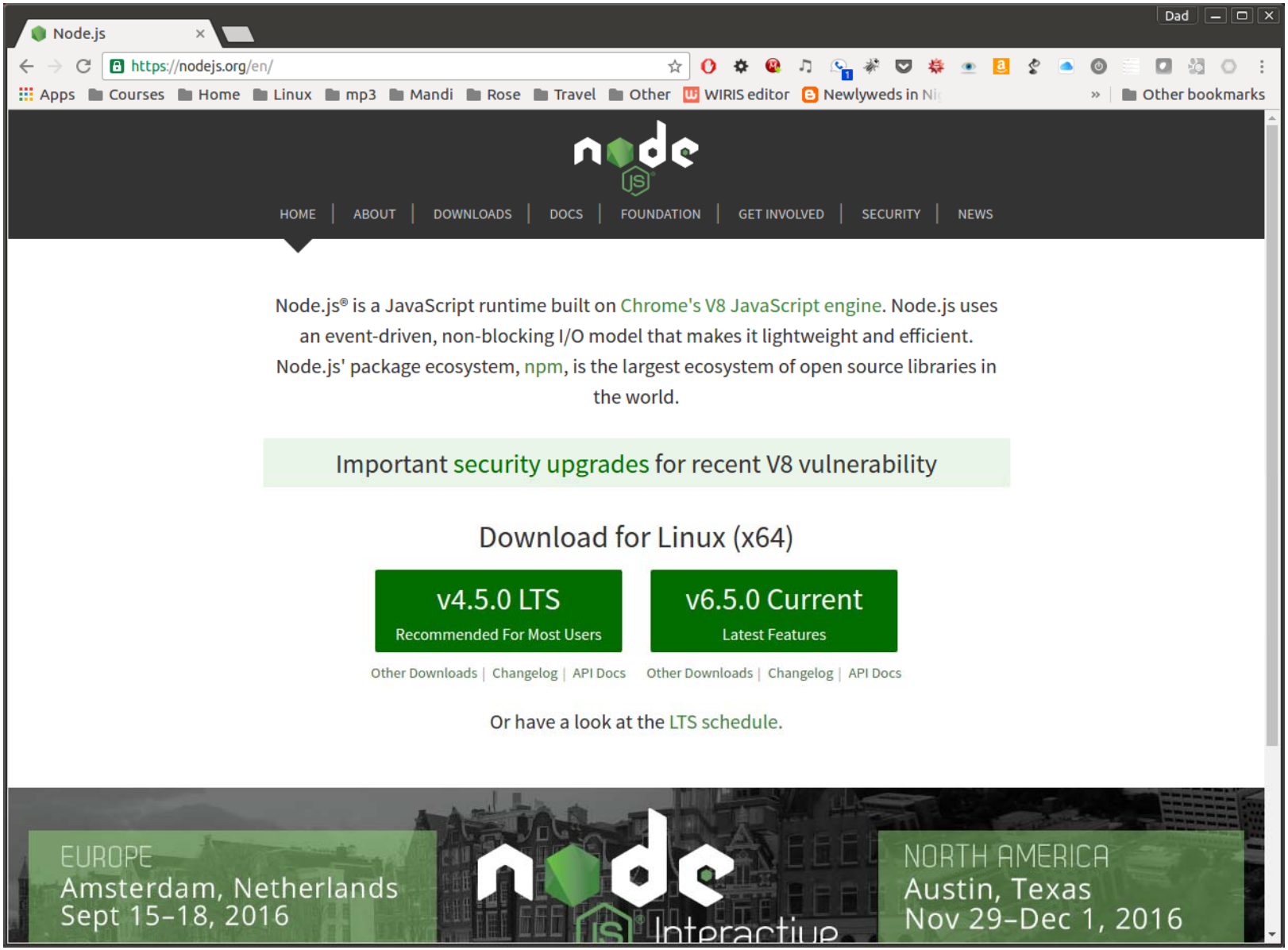
- JavaScript is the programming language of HTML and the Web. (http://www.w3schools.com/js/)
- http://stackoverflow.com/questions/245062/whats-the-difference-between-javascript-and-java



node.js

- Platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications.
- Uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.
- Programmed in JavaScript on both server and client.

http://nodejs.org/


Node.js
Dad
https://nodejs.org/en/
Apps
Courses
Home
Linux
mp3
Mandi
Rose
Travel
Other
WIRIS editor
Newlyweds in Ni
Other bookmarks
HOME
ABOUT
DOWNLOADS
DOCS
FOUNDATION
GET INVOLVED
SECURITY
NEWS
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.
Important security upgrades for recent V8 vulnerability
Download for Linux (x64)
v4.5.0 LTS
Recommended For Most Users
v6.5.0 Current
Latest Features
Other Downloads | Changelog | API Docs
Other Downloads | Changelog | API Docs
Or have a look at the LTS schedule.
EUROPE
Amsterdam, Netherlands
Sept 15–18, 2016
Interactive
NORTH AMERICA
Austin, Texas
Nov 29–Dec 1, 2016

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Important security releases, please update now!

Download for Windows (x64)

v6.11.3 LTS
Recommended For Most Users

Other Downloads | Changelog | API Docs

v8.5.0 Current
Latest Features

Other Downloads | Changelog | API Docs

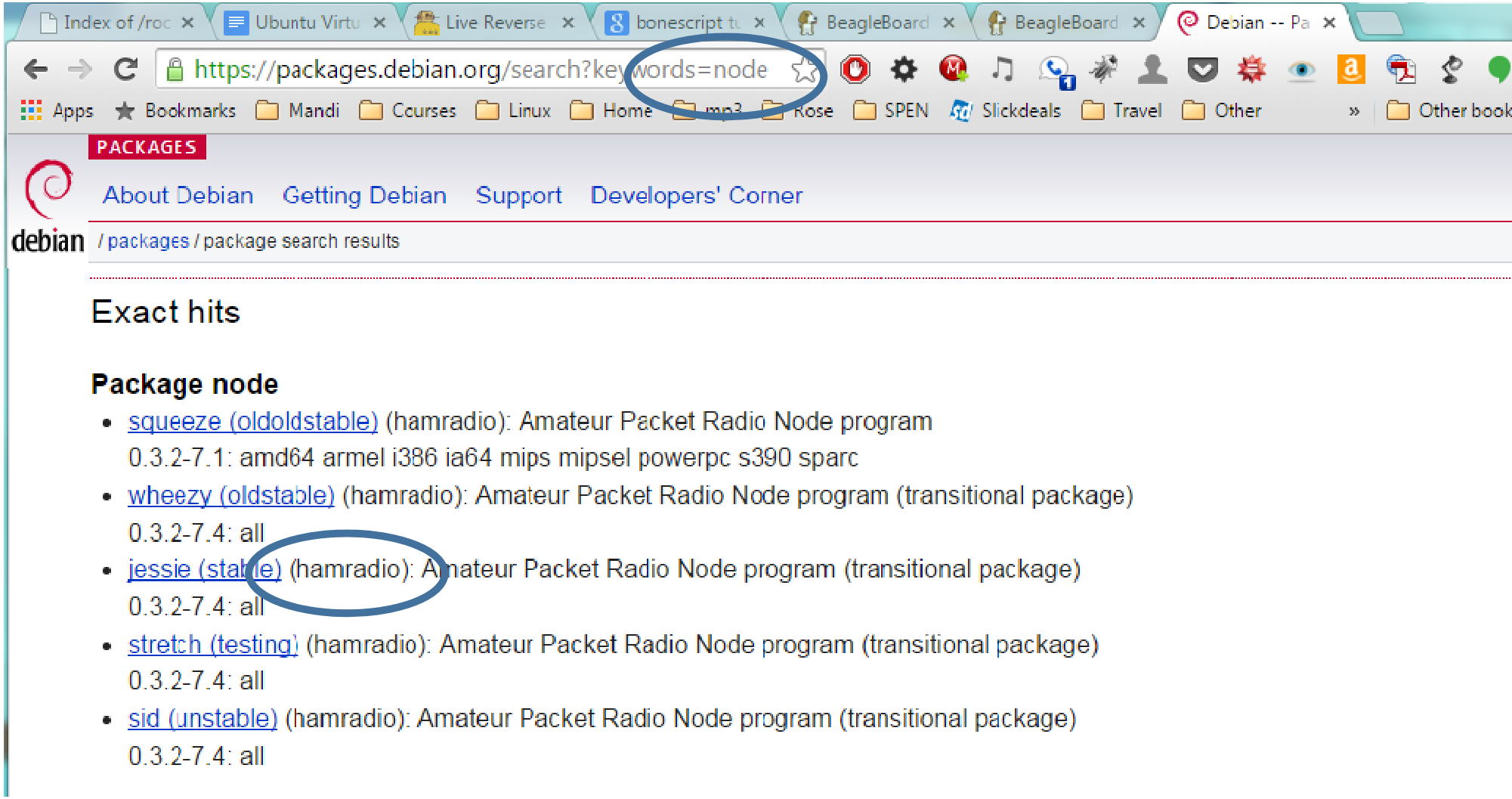
Installing node.js

- Already installed on Bone, but you can update it with:

```
bone$ sudo apt install nodejs
```

- You can also install it on your host, but watchout:

```
host$ sudo apt install nodejs
host$ cd /usr/bin
host$ sudo ln -s nodejs node
```


Index of /roc
Ubuntu Virtu
Live Reverse
bonescript tu
BeagleBoard
BeagleBoard
Debian -- Pa
https://packages.debian.org/search?keywords=node
Apps
Bookmarks
Mandi
Courses
Linux
Home
Rose
SPEN
Slickdeals
Travel
Other
Other book
PACKAGES
About Debian
Getting Debian
Support
Developers' Corner
debian
/ packages / package search results
Exact hits
Package node
squeeze (oldoldstable) (hamradio): Amateur Packet Radio Node program
0.3.2-7.1: amd64 armel i386 ia64 mips mipsel powerpc s390 sparc
wheezy (oldstable) (hamradio): Amateur Packet Radio Node program (transitional package)
0.3.2-7.4: all
jessie (stable) (hamradio): Amateur Packet Radio Node program (transitional package)
0.3.2-7.4: all
stretch (testing) (hamradio): Amateur Packet Radio Node program (transitional package)
0.3.2-7.4: all
sid (unstable) (hamradio): Amateur Packet Radio Node program (transitional package)
0.3.2-7.4: all

node.js example: Webserver

- This simple web server written in Node responds with “Hello World” for every request.

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337);
console.log('Server running on port 1337');
```

- To run the server, put the code into a file *example.js* and execute it with the node program:

```
$ node example.js
Server running on port 1337
```

JavaScript – C-like

```
#include <stdio.h>
main() {
    int i;
    for(i=0; i<5; i++) {
       printf("i=%d\n", i);
    }
}
```

```
var i;
for(i=0; i<5; i++) {
    console.log("i=%d", i);
}
```

JavaScript in 10 minutes

- By Spencer Tipping
  - https://github.com/spencertipping/js-in-ten-minutes
  - 27 pages
- **OR**
- https://sites.google.com/site/solopurotutoriales/javascript-in-ten-minutes
  - **9 pages**
  - Here are the highlights…

JS - Types

- **Strings** – e.g. 'foo', "foo" (single vs. double quotation – no difference)
- **Numbers** – e.g. 5, 3e+10 (all numbers behave as floats)
- **Booleans** – true and false
- **Arrays** – e.g. **[**1, 2, "foo", [3, 4]**]**
- **Objects** – e.g. {foo**:** 'bar', bif**:** [1, 2]}, which are really just hashtables
- **Functions** – e.g. var example=function (x) {return x + 1}

JS - Functions

- Functions are first-class lexical closures

```
var f = function () { // f is toplevel, so global
    var x = 5;        // x is local to f
    y = 6;            // y is global
};
```

- Watch out

```
var f = function () { // f is toplevel, so global
    y = 6;            // y is global
    x = 42;
Do stuff…
    var x = 5;        // x is local to f
};
```

JS - Semicolon

- JavaScript doesn't require a semicolon at the end of each line, but you should anyway.

```
var x = f
(y = x) (5)
```

- Is treated as:

```
var x = f(y = x) (5)
```

- You probably meant

```
var x = f;
(y = x) (5);
```

JS - Equality

- Never use **==** or **!=**
- Always use **===** or **!==**
- All these are **true**:

```
null  == undefined
null  == 0
false == ''
''    == 0
true  == 1
true  == '1'
```

JavaScript: The Good Parts

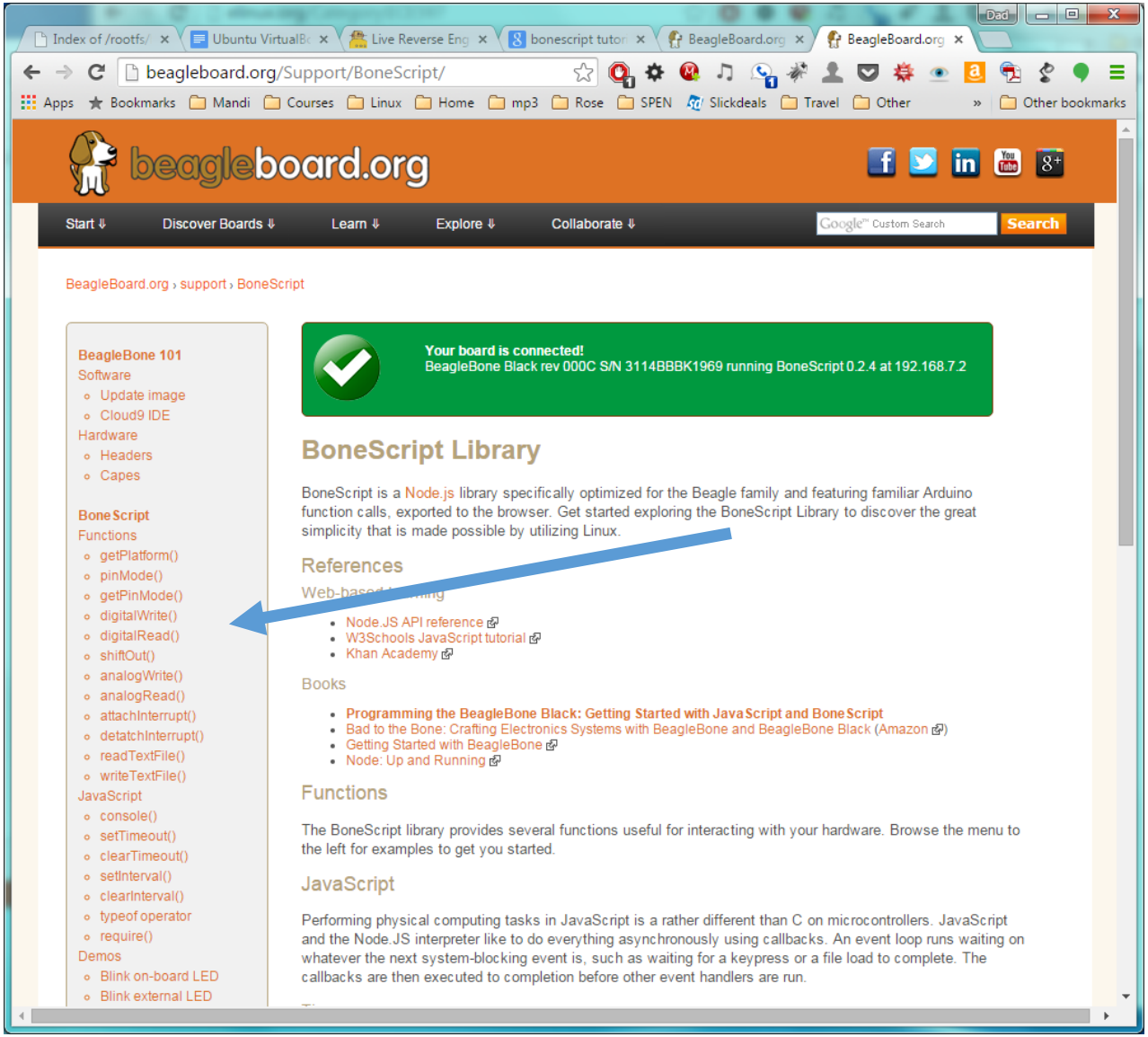
- Intended for programmers who, by happenstance or curiosity, are venturing into JavaScript.
- Also intended for programmers who have been working with JavaScript at a novice level and are now ready for a more sophisticated relationship with the language.
- Most programming languages contain **good parts and bad parts**. I discovered that I could be a better programmer by using only the good parts and avoiding the bad parts.
- JavaScript is a language with more than its share of bad parts.
- 172 pages

JavaScript Differences

- Performing physical computing tasks in JavaScript is a rather different than C on microcontrollers
- JavaScript and the Node.JS interpreter like to do everything *asynchronously* using *callbacks*
- An event loop runs waiting on whatever the next system-blocking event is,
  - such as waiting for a keypress or a file load to complete
- The callbacks are then executed to completion before other event handlers are run

BoneScript

- “BoneScript is a JavaScript library to simplify learning how to perform physical computing tasks using your embedded Linux”
- http://beagleboard.org/support/bone101
- “BoneScript is a Node.js library specifically optimized for the Beagle family and featuring familiar Arduino function calls, exported to the browser.”
- http://beagleboard.org/Support/BoneScript/


Index of /rootfs/
Ubuntu VirtualBo
Live Reverse Eng
bonescript tutor
BeagleBoard.org
BeagleBoard.org
beagleboard.org/Support/BoneScript/
Apps
Bookmarks
Mandi
Courses
Linux
Home
mp3
Rose
SPEN
Slickdeals
Travel
Other
Other bookmarks
beagleboard.org
Start
Discover Boards
Learn
Explore
Collaborate
Search
BeagleBoard.org › support › BoneScript
BeagleBone 101
Software
Update image
Cloud9 IDE
Hardware
Headers
Capes
BoneScript
Functions
getPlatform()
pinMode()
getPinMode()
digitalWrite()
digitalRead()
shiftOut()
analogWrite()
analogRead()
attachInterrupt()
detatchInterrupt()
readTextFile()
writeTextFile()
JavaScript
console()
setTimeout()
clearTimeout()
setInterval()
clearInterval()
typeof operator
require()
Demos
Blink on-board LED
Blink external LED
Your board is connected!
BeagleBone Black rev 000C S/N 3114BBBK1969 running BoneScript 0.2.4 at 192.168.7.2
BoneScript Library
BoneScript is a Node.js library specifically optimized for the Beagle family and featuring familiar Arduino function calls, exported to the browser. Get started exploring the BoneScript Library to discover the great simplicity that is made possible by utilizing Linux.
References
Node.JS API reference
W3Schools JavaScript tutorial
Khan Academy
Books
Programming the BeagleBone Black: Getting Started with JavaScript and BoneScript
Bad to the Bone: Crafting Electronics Systems with BeagleBone and BeagleBone Black (Amazon)
Getting Started with BeagleBone
Node: Up and Running
Functions
The BoneScript library provides several functions useful for interacting with your hardware. Browse the menu to the left for examples to get you started.
JavaScript
Performing physical computing tasks in JavaScript is a rather different than C on microcontrollers. JavaScript and the Node.JS interpreter like to do everything asynchronously using callbacks. An event loop runs waiting on whatever the next system-blocking event is, such as waiting for a keypress or a file load to complete. The callbacks are then executed to completion before other event handlers are run.