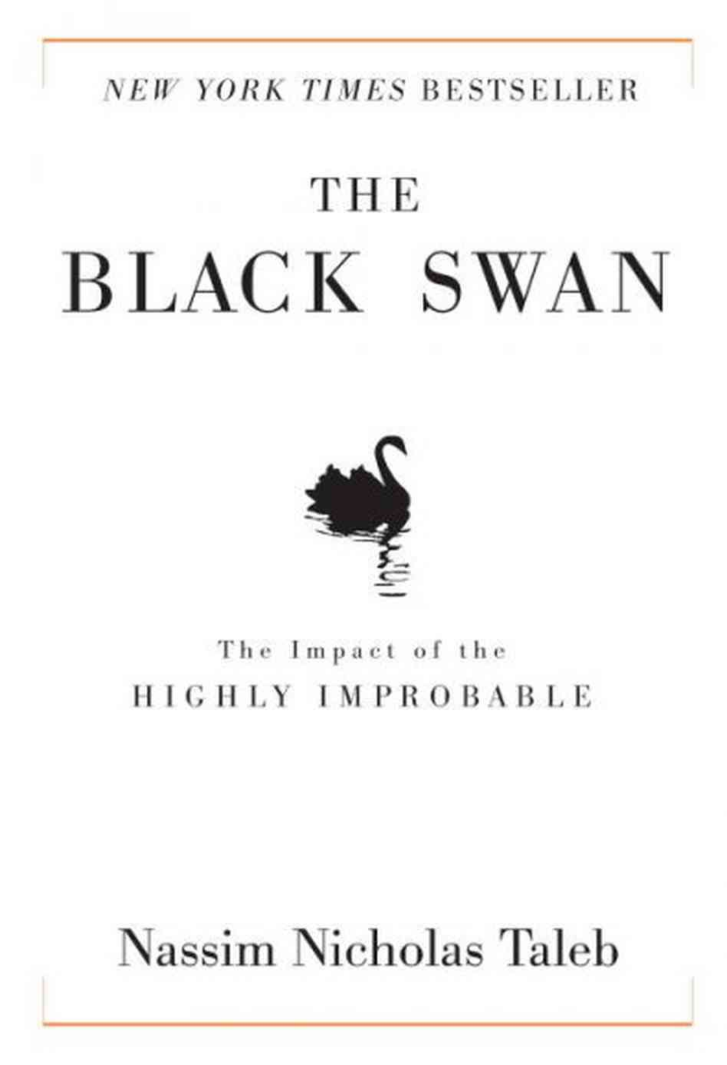
Javascript

it’s reasonable



NEW YORK TIMES BESTSELLER

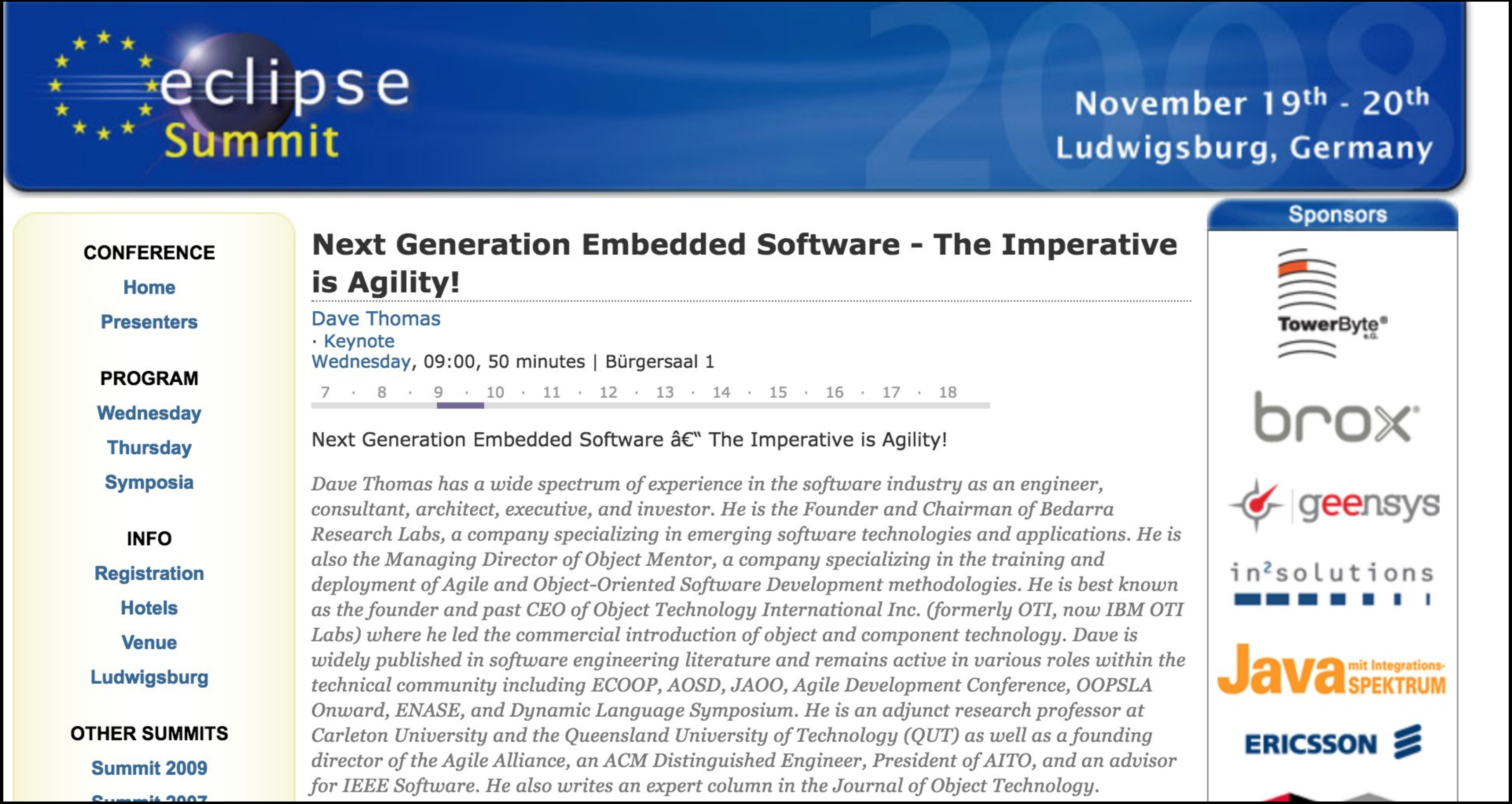
THE BLACK SWAN

The Impact of the HIGHLY IMPROBABLE

Nassim Nicholas Taleb

LiveScript ‘95

F*** You! Money



Next Generation Embedded Software - The Imperative is Agility!

Dave Thomas

· Keynote

Wednesday, 09:00, 50 minutes | Bürgersaal 1

Next Generation Embedded Software â€" The Imperative is Agility!

Dave Thomas has a wide spectrum of experience in the software industry as an engineer, consultant, architect, executive, and investor. He is the Founder and Chairman of Bedarra Research Labs, a company specializing in emerging software technologies and applications. He is also the Managing Director of Object Mentor, a company specializing in the training and deployment of Agile and Object-Oriented Software Development methodologies. He is best known as the founder and past CEO of Object Technology International Inc. (formerly OTI, now IBM OTI Labs) where he led the commercial introduction of object and component technology. Dave is widely published in software engineering literature and remains active in various roles within the technical community including ECOOP, AOSD, JAOO, Agile Development Conference, OOPSLA Onward, ENASE, and Dynamic Language Symposium. He is an adjunct research professor at Carleton University and the Queensland University of Technology (QUT) as well as a founding director of the Agile Alliance, an ACM Distinguished Engineer, President of AITO, and an advisor for IEEE Software. He also writes an expert column in the Journal of Object Technology.

Don’t
(always)
believe
me

Systems	Application

Systems

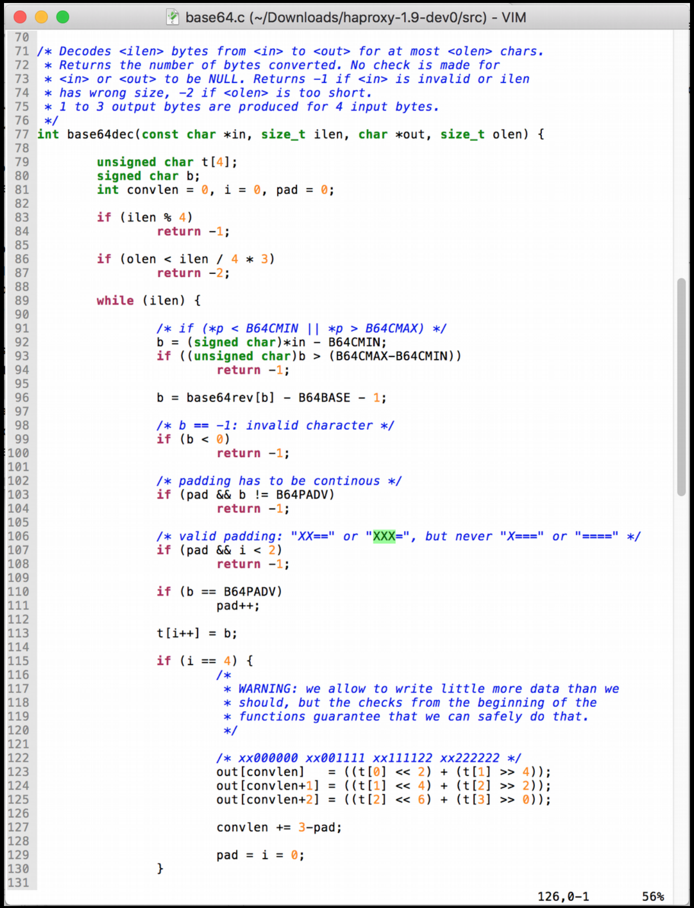
- operating systems
- database engines
- file servers
- device drivers
- routers
- browsers

Application

- anything that answers business requirements

Systems	Application
• digital capability	• business value
• computer science	• computer engineering
• write for machines	• write for people
• slow, painstaking	• fast, replaceable / updatable
• optimisation: byte and cycle level	• optimisation: IO and algorithmic
• lifetime: decades	• lifetime: years
• costs up front	• ongoing costs

Systems

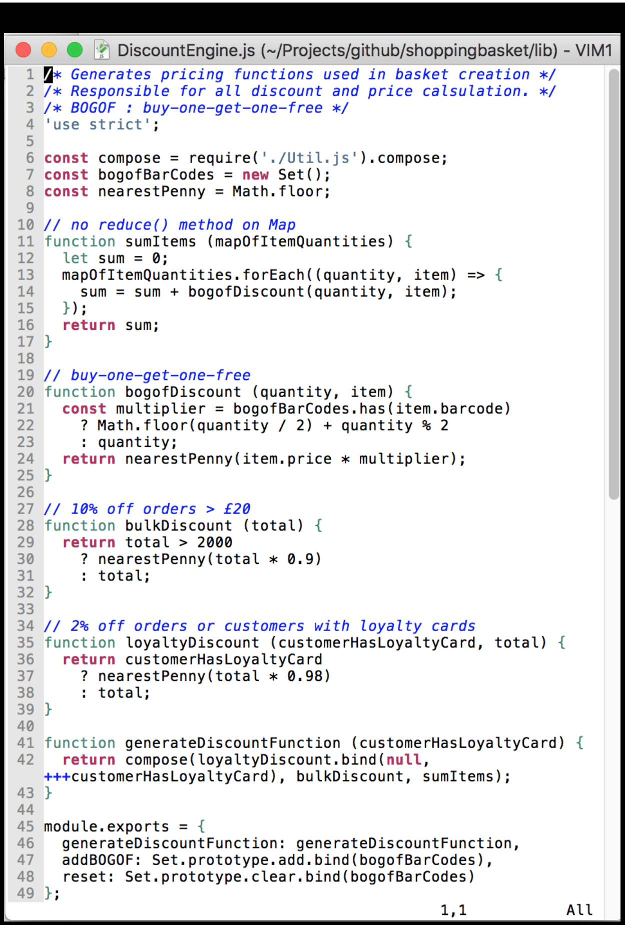


base64.c (~/Downloads/haproxy-1.9-dev0/src) - VIM

```c

/* Decodes <ilen> bytes from <in> to <out> for at most <olen> chars.
 * Returns the number of bytes converted. No check is made for
 * <in> or <out> to be NULL. Returns -1 if <in> is invalid or ilen
 * has wrong size, -2 if <olen> is too short.
 * 1 to 3 output bytes are produced for 4 input bytes.
 */
int base64dec(const char *in, size_t ilen, char *out, size_t olen) {

        unsigned char t[4];
        signed char b;
        int convlen = 0, i = 0, pad = 0;

        if (ilen % 4)
                return -1;

        if (olen < ilen / 4 * 3)
                return -2;

        while (ilen) {

                /* if (*p < B64CMIN || *p > B64CMAX) */
                b = (signed char)*in - B64CMIN;
                if ((unsigned char)b > (B64CMAX-B64CMIN))
                        return -1;

                b = base64rev[b] - B64BASE - 1;

                /* b == -1: invalid character */
                if (b < 0)
                        return -1;

                /* padding has to be continous */
                if (pad && b != B64PADV)
                        return -1;

                /* valid padding: "XX==" or "XXX=", but never "X===" or "====" */
                if (pad && i < 2)
                        return -1;

                if (b == B64PADV)
                        pad++;

                t[i++] = b;

                if (i == 4) {
                        /*
                         * WARNING: we allow to write little more data than we
                         * should, but the checks from the beginning of the
                         * functions guarantee that we can safely do that.
                         */

                        /* xx000000 xx001111 xx111122 xx222222 */
                        out[convlen]   = ((t[0] << 2) + (t[1] >> 4));
                        out[convlen+1] = ((t[1] << 4) + (t[2] >> 2));
                        out[convlen+2] = ((t[2] << 6) + (t[3] >> 0));

                        convlen += 3-pad;

                        pad = i = 0;
                }

```

126,0-1   56%

Application



DiscountEngine.js (~/Projects/github/shoppingbasket/lib) - VIM1

```javascript
/* Generates pricing functions used in basket creation */
/* Responsible for all discount and price calsulation. */
/* BOGOF : buy-one-get-one-free */
'use strict';

const compose = require('./Util.js').compose;
const bogofBarCodes = new Set();
const nearestPenny = Math.floor;

// no reduce() method on Map
function sumItems (mapOfItemQuantities) {
  let sum = 0;
  mapOfItemQuantities.forEach((quantity, item) => {
    sum = sum + bogofDiscount(quantity, item);
  });
  return sum;
}

// buy-one-get-one-free
function bogofDiscount (quantity, item) {
  const multiplier = bogofBarCodes.has(item.barcode)
    ? Math.floor(quantity / 2) + quantity % 2
    : quantity;
  return nearestPenny(item.price * multiplier);
}

// 10% off orders > £20
function bulkDiscount (total) {
  return total > 2000
    ? nearestPenny(total * 0.9)
    : total;
}

// 2% off orders or customers with loyalty cards
function loyaltyDiscount (customerHasLoyaltyCard, total) {
  return customerHasLoyaltyCard
    ? nearestPenny(total * 0.98)
    : total;
}

function generateDiscountFunction (customerHasLoyaltyCard) {
  return compose(loyaltyDiscount.bind(null,
+++customerHasLoyaltyCard), bulkDiscount, sumItems);
}

module.exports = {
  generateDiscountFunction: generateDiscountFunction,
  addBOGOF: Set.prototype.add.bind(bogofBarCodes),
  reset: Set.prototype.clear.bind(bogofBarCodes)
};
```

1,1   All

Systems

- C (++)

Application

- Java
- Python
- Ruby
- Javascript
- etc.

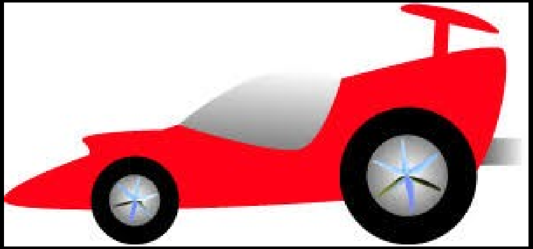
Javascript Pro’s

- easy to learn, not too hard to master
- dynamic, i.e. not static
- single threaded, run-to-completion
- no exposing OS primitives
- supports a variety of styles
  - imperative / functional
  - classical / prototypical inheritance
  - callbacks / continuation passing style / es6

Javascript

it’s reasonable

Speed



(almost) irrelevant

Speaking Intelligently about "Java vs Node" Performance

Please understand what's really important when it comes to the benefits of a platform.

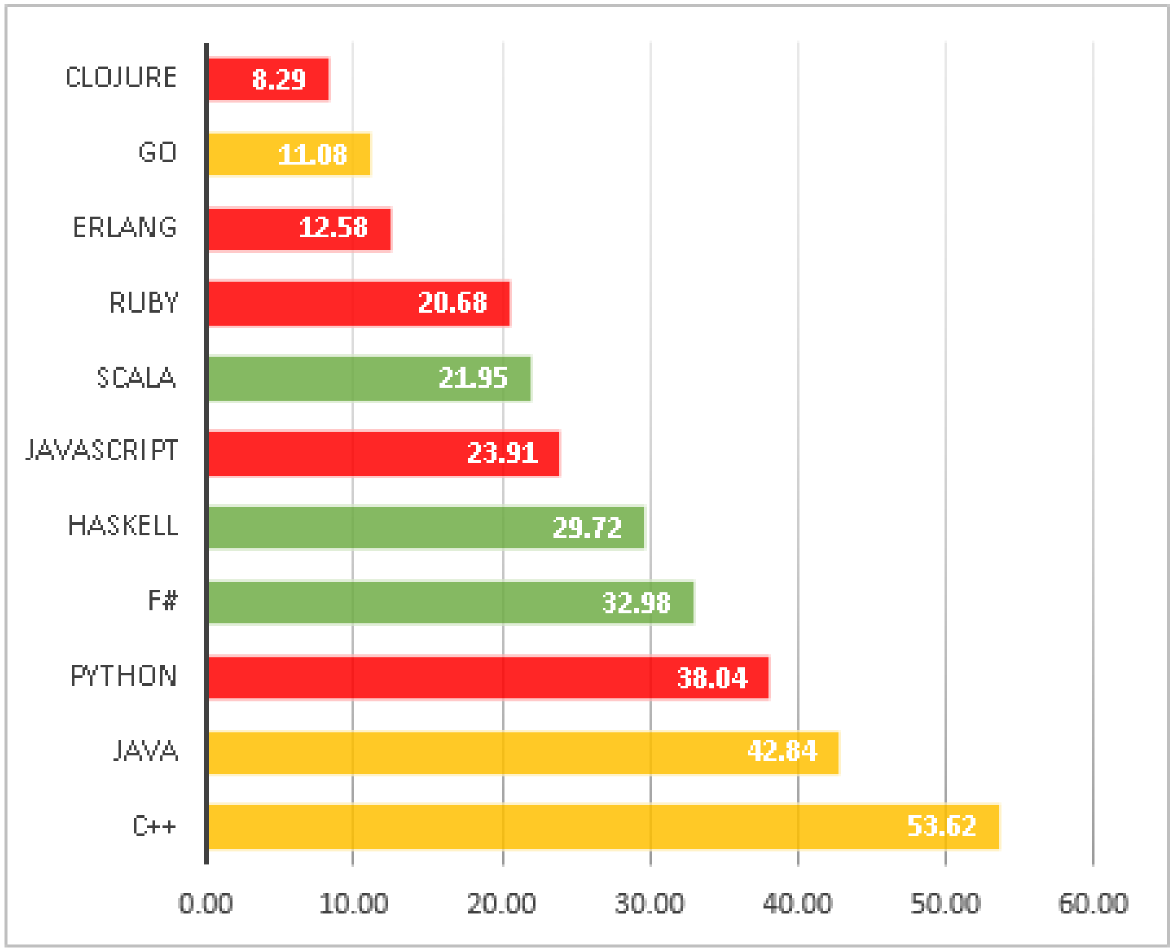
Richard Clayton

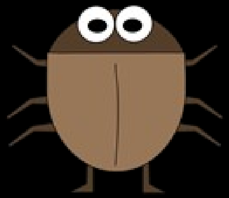
Unrepentant Thoughts on Software and Management.

https://rclayton.silvrback.com/speaking-intelligently-about-java-vs-node-performance

Bugs

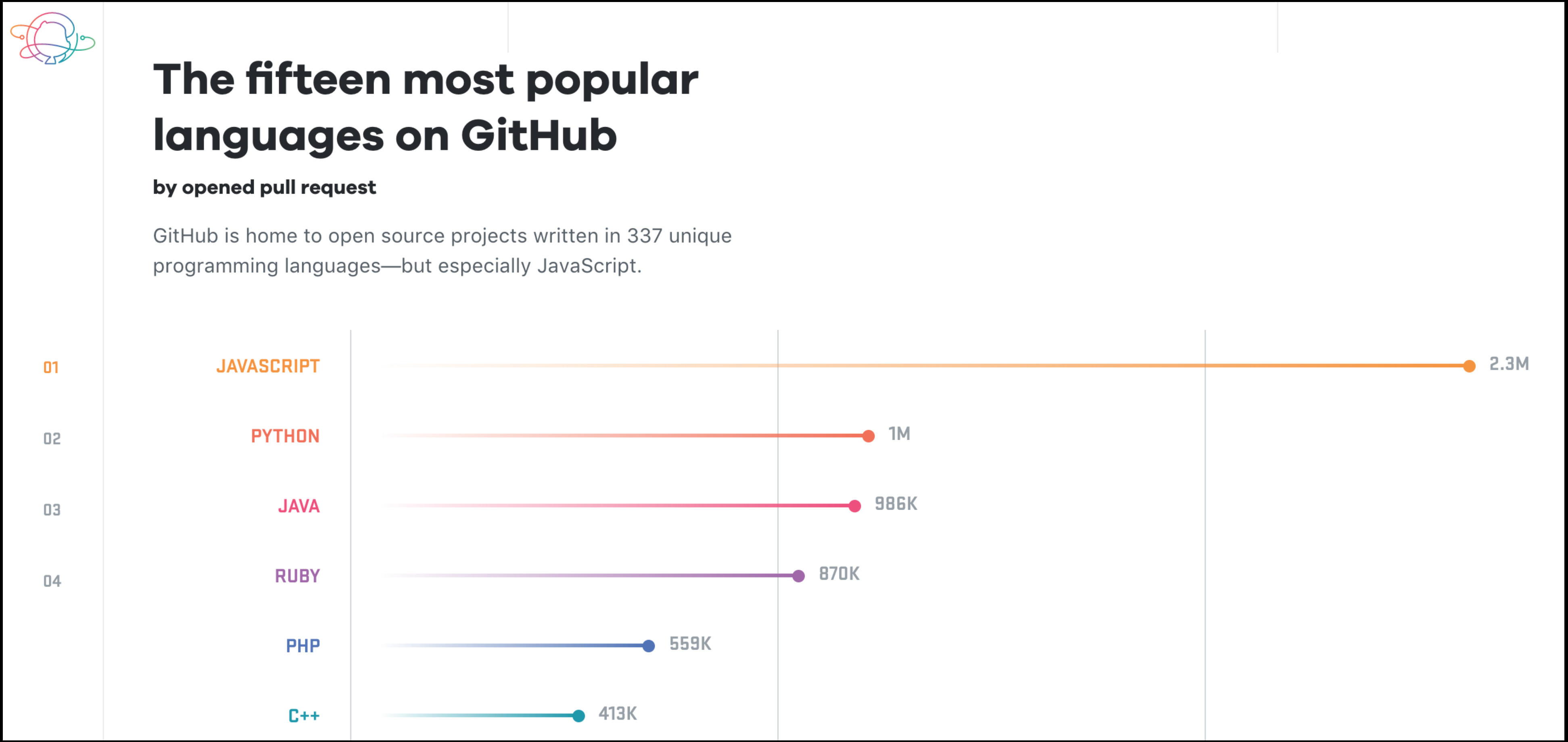
Round 3. Languages sorted by bug density. More than 100 stars repos





https://labs.ig.com/static-typing-promise

Popularity

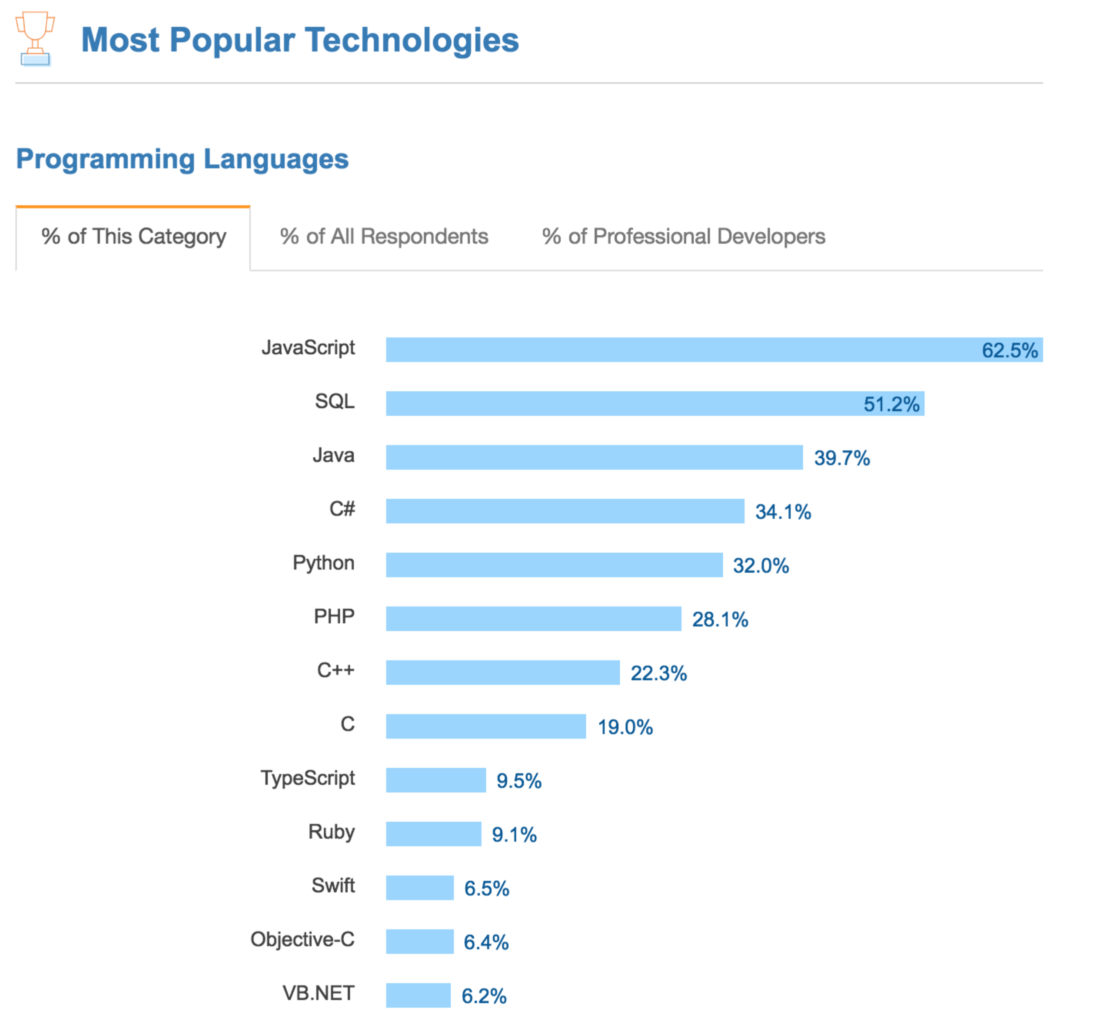


The fifteen most popular languages on GitHub

by opened pull request

GitHub is home to open source projects written in 337 unique programming languages—but especially JavaScript.

https://octoverse.github.com/



Most Popular Technologies

Programming Languages

https://insights.stackoverflow.com/survey/2017#most-popular-technologies

Systems	Application
C (++)	JS

Session 2

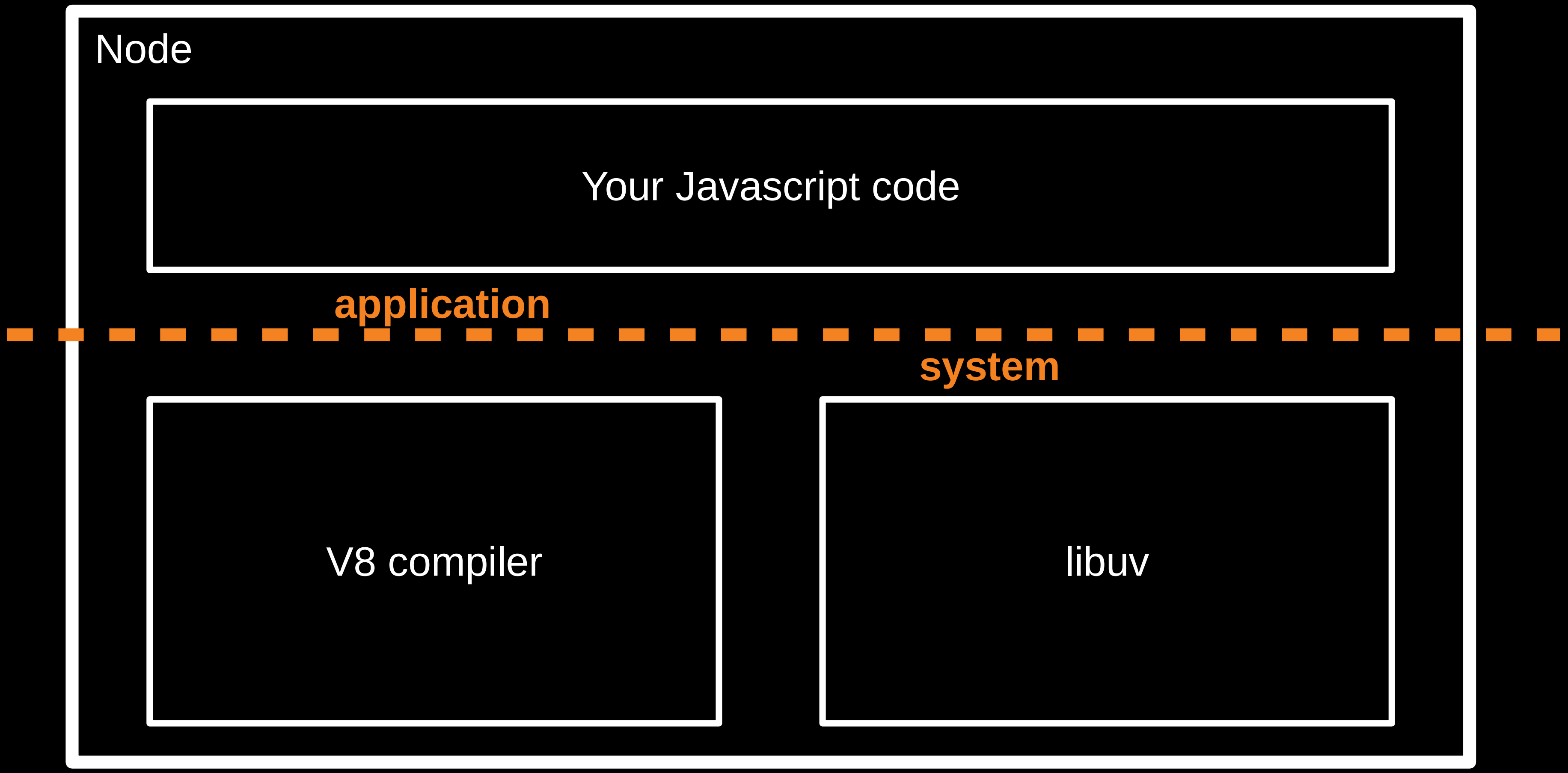
Monday 12th March 12:45

Javascript: its natural habitat - the browser

Session 3

Monday 26th March 12:45

Javascript: its new natural habitat - everywhere



Node

Your Javascript code

application

system

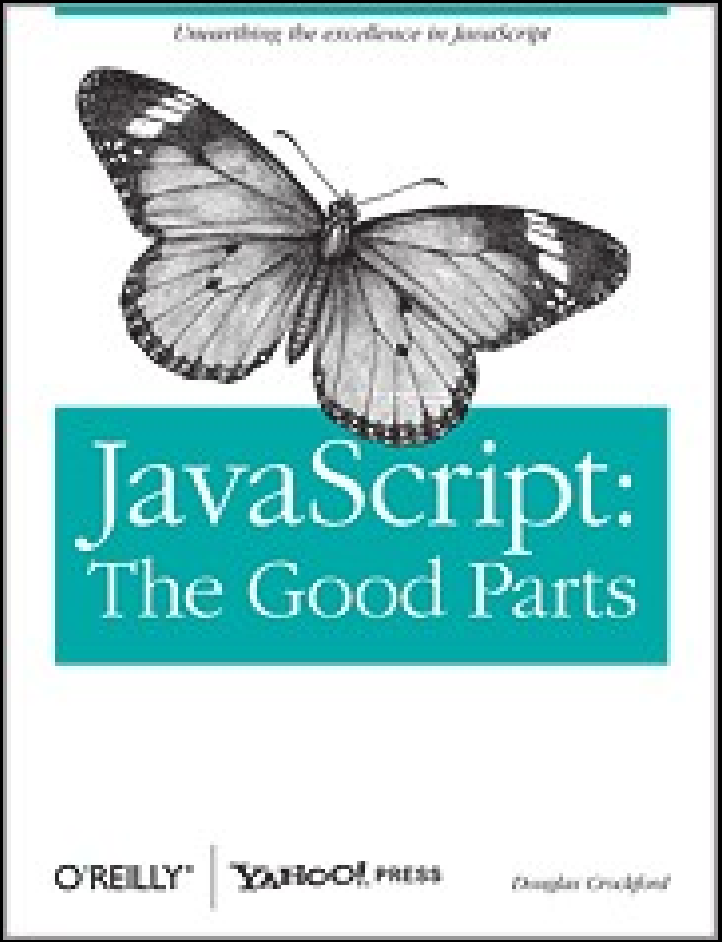
V8 compiler

libuv

Session 4

Monday 2nd April 12:30

Javascript for Java developers

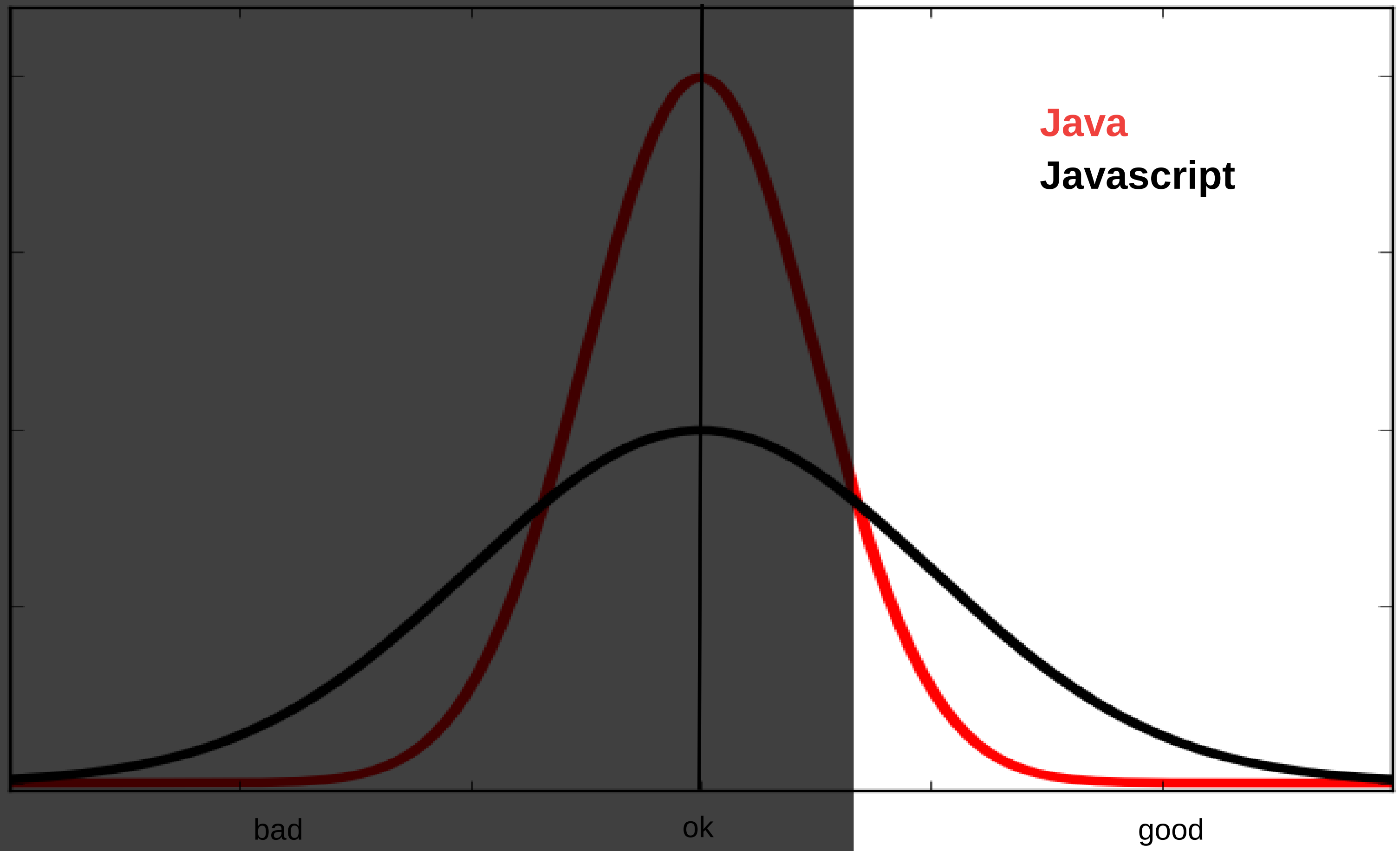


Unearthing the excellence in JavaScript

JavaScript: The Good Parts

O'REILLY® | YAHOO! PRESS

Douglas Crockford


Java
Javascript
bad
ok
good

Session 5 - Workshop

Monday 9th April 12:30

A Scary Walking Skeleton

Pre-workshop preparation

- ensure your computer has
  - git
  - node
  - access to:
    - https://jsbin.com/
    - https://github.com/
- or find a coding buddy who does

Session 6 : Workshop

Monday 16th April 12:30

Grown up Javascript – the enterprise era

Thank You

https://www.surveymonkey.co.uk/r/79RVJWW