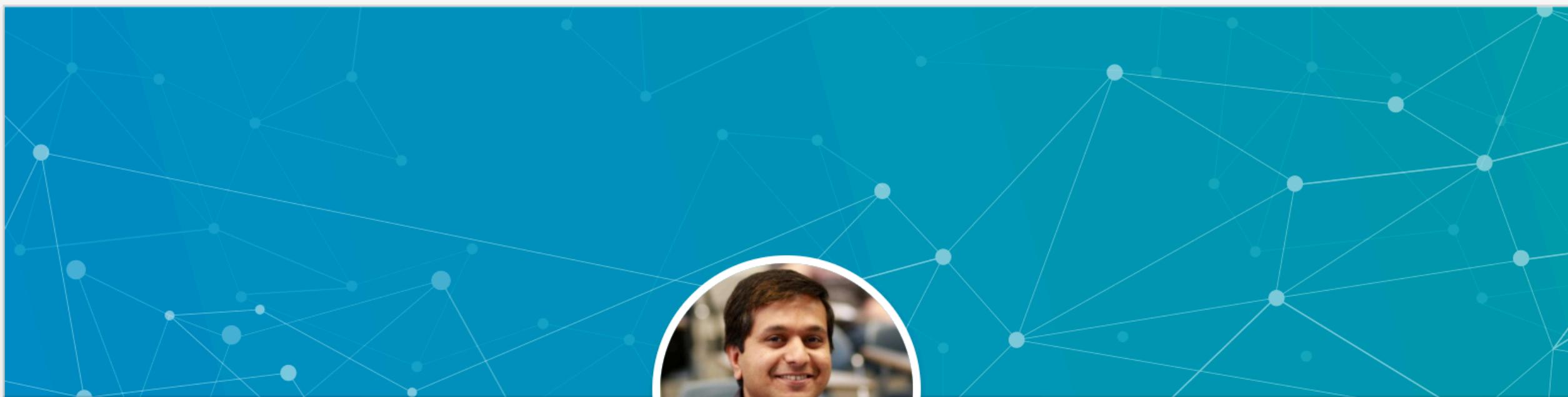


ReasonML

IndyJS, August 2017

Pradeep Gowda <@btbytes>



Pradeep Kishore Gowda

Staff Engineer at Proofpoint

Proofpoint • Indiana University–Purdue University Indianapolis

Indianapolis, Indiana Area • 167 

<https://www.linkedin.com/in/btbytes/>

Q&A

**How many program in javascript
regularly?**

What is the biggest Javascript codebase you have worked with:

- 1000 lines
- 5000 lines
- 10,000 lines

**How many "program in more than
3 programming languages on a
regular basis"**

**How many of you have heard/
programmed in ReasonML?**

**How many of you know/program
using React?**

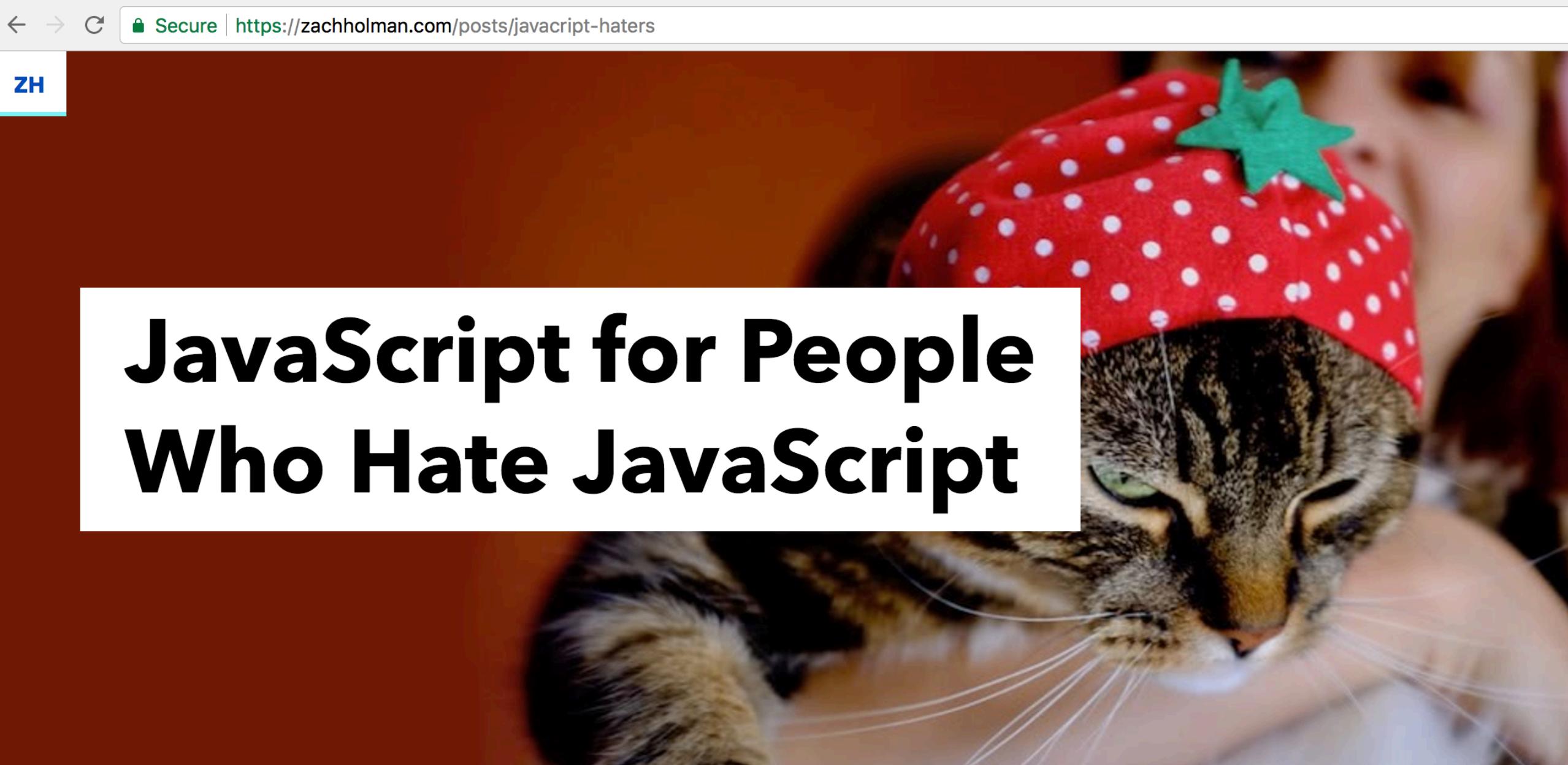
**The phrase "Static Typing"
means ...**

**The phrase "Functional
Programming" means ...**

Tests remove the need for static types ...

1. yes
2. OF COURSE!
3. no
4. maybe

[^_ ^]



JavaScript for People Who Hate JavaScript

August 8, 2017

I have a long history with JavaScript, dating back to the glory days of the most

Zach Holman

Complaints

"there are five hundred thousand packages to capitalize the first letter of every word, and each of them have about fifty hundred thousand dependencies, each of which is in turn has about one line of code."

etc., etc.,

-- JavaScript for People Who Hate JavaScript

React is cool, OMG

I found CRA fairly mind-blowing, because it took care of all of these fairly answerable questions for me: how do I bundle my code? How do I structure my code? How do I do hot code reloading (which is a great name for a band)? How about code splitting? How do I set up testing? How do I work with Babel, whatever the fuck *that* is? (Just kidding. I know what it is by now. It's terrifying, that's what it is.) These are all answerable questions, but I just don't have the time to spend months and months getting to know all the options, trying them out, and getting them all to work together. CRA has best-in-industry people like Dan Abramov doing all this crazy shit for me.

JavaScript always seemed a little bonkers. From having a pretty terrible standard library, to callback hell, it always seemed weird to use, whether I was using jQuery or vanilla JavaScript.

Couple new things have happened in JavaScript land that have changed my opinion: prettier, and the ability to bring in ES6 and ES7 features, which makes JavaScript the language itself much more palatable. Create React App helps a ton with the latter; they already bring in a few polyfills, and the way they've set up your toolchain makes it easier to bring in additional forward-looking JavaScript modules and libraries.

Prettier, if you haven't heard of it, is awesome. A screencast says about a billion words:

Static Typing

Functional Programming

**Tools to improve
Javascript experience**

Flow

Flow is a static typechecker for Javascript

Flow is a static type checker for your JavaScript code. It does a lot of work to make you more productive. Making you code faster, smarter, more confidently, and to a bigger scale.

Flow checks your code for errors through **static type annotations**. These types allow you to tell Flow how you want your code to work, and Flow will make sure it does work that way.

```
1 // @flow
2 function square(n: number): number {
3   string This type is incompatible with the expected param type of number
4
5
6 square("2"); // Error!
```

Because Flow understands JavaScript so well, it doesn't need many of these types. You should only ever have to do a minimal amount of work to describe your code to Flow and it will *infer* the rest. A lot of the time, Flow can understand your code without any types at all.

```
1 // @flow
2 function square(n) {
3   return n * n; // Error!
4 }
5
6 square("2");
```

Flow

Prettier

"Opinionated Code Formatter" for javascript

BabelJS

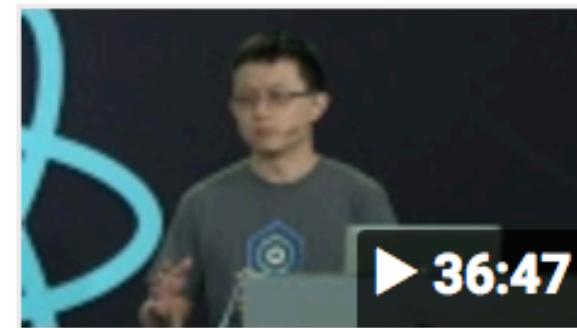
"The compiler for writing next generation JavaScript"

Google Closure

"The Closure Compiler compiles JavaScript into compact, high-performance code. The compiler removes dead code and rewrites and minimizes what's left so that it downloads and runs quickly. It also checks syntax, variable references, and types, and warns about common JavaScript pitfalls. These checks and optimizations help you write apps that are less buggy and easier to maintain."

Taming the meta language

Cheng Lou - Taming the Meta Language - React Conf 2017 - YouTube



https://www.youtube.com/watch?v=_0T5OSSzxms

Mar 16, 2017 - Uploaded by Facebook Developers

Cheng Lou - Taming the Meta Language - React Conf 2017. Facebook Developers.

Loading... Unsubscribe ...

-- <@_chenglou>

conferences
meetups
books
online forums
blog posts
videos
tutorials
examples
documentation
version control
editor assistance
comments
tests
code (language)

▶ ▶! 🔍 5:06 / 36:46



Cheng Lou - Taming the Meta Language - React Conf 2017



Facebook Developers

Subscribe 92K

10,416 views

Compile to JS

- what are you talking about?
- I don't like them because ____
- I have used them, but ____

List of languages that
compile to JS .

jashkenas/
coffeescript Wiki

Scala.js

Feature	JavaScript ES5	JavaScript ES6	TypeScript	Scala.js
Interoperability				
Fully EcmaScript5 compatible	●	●	●	●
No compilation required	●	○	○	○
Use existing JS libraries	●	●	●	●
Language features				
Classes with inheritance	○	●	●	●
Modules	○	●	●	●
Support for types	○	○	●	●
Strong type system	○	○	○	●
Extensive standard libraries	○	○	○	●
Optimizing compiler	○	○	○	●
Macros, to extend the language	○	○	○	●
IDE support				
Catch most errors in IDE	○	○	●	●
Easy and reliable refactoring	○	○	●	●
Reliable code completion	○	○	●	●

More mainstream(?)

- Dart
- TypeScript

More fun(?)

- Elm --> ML/Haskellish. UI/Gaming focussed.
- Purescript --> Haskellish language; multiple backends
- Idris --> A Dependently Typed Functional Programming Language (Haskell/MLish)

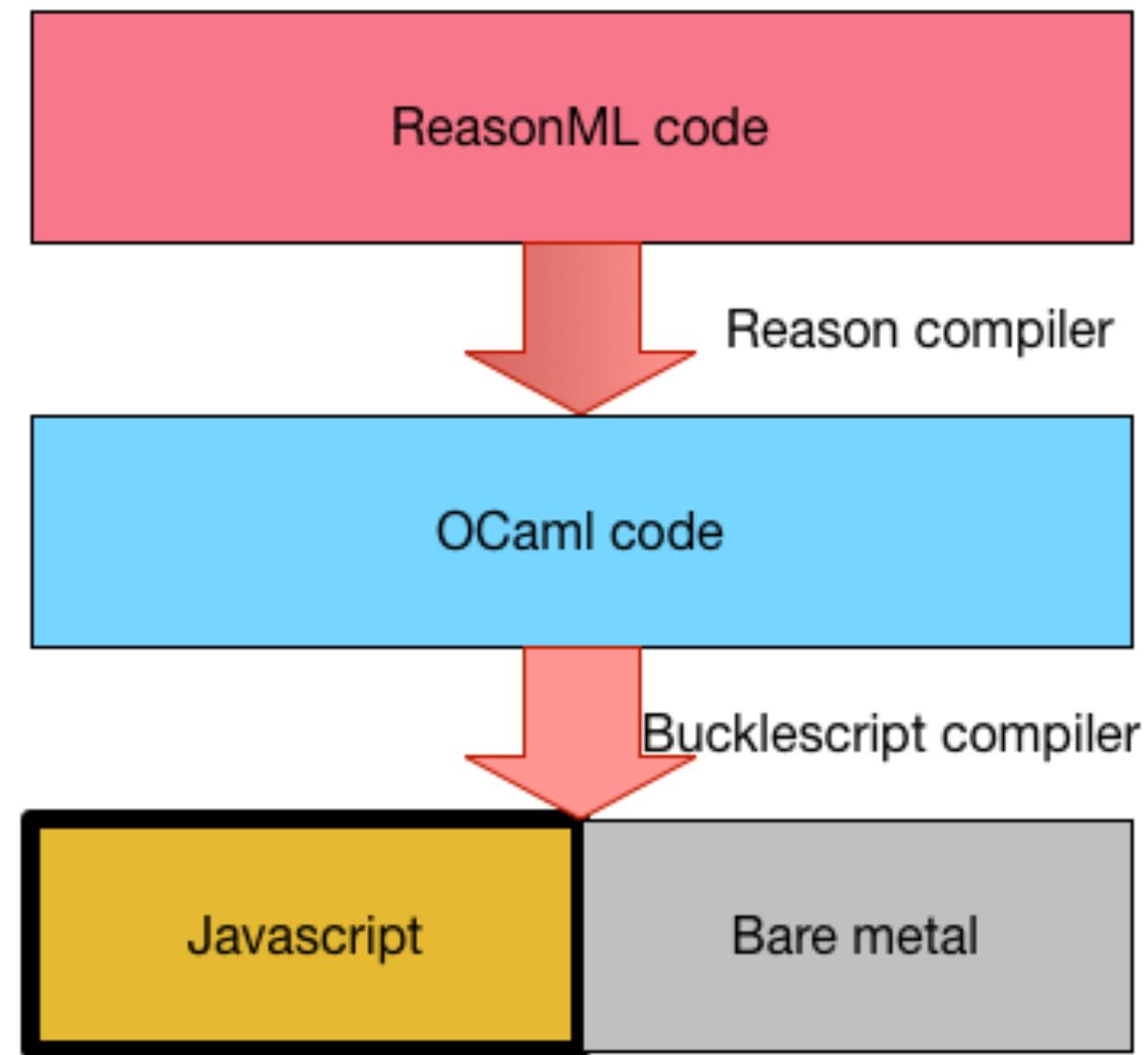
- C/C++ --> Emscripten; compile to asm.js
- **bucklescript** --> OCaml

What is ReasonML

- Syntax and toolchain for OCaml
- Friendly toolchain on top of OCaml
- From the creator of React
- make it familiar to the Javascript developers.



ReasonML compilation chain



OCaml

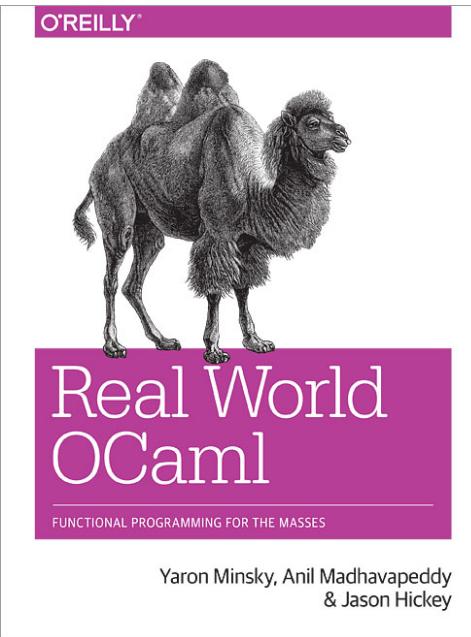
- Originally known as Objective Caml
- a derivative of ML with OOP extensions
- 20 years. Made in France
- has roots in ML
- ... see also SML, F#

OCaml

- Known for ...see list...
- Operating systems
- Compilers
- Formal verification tools (eg: Compcert)
- Static Code analysis (eg: Frama-C)
- Coq - formal proof management system.



Real World OCaml



- O'Reilly book
- Free to read online
- website

OCaml in the "news"

Hack PHP compiler

Hack

```
1 <?hh
2 class MyClass {
3     public function alpha(): int {
4         return 1;
5     }
6
7     public function beta(): string {
8         return 'hi test';
9     }
10}
11
12 function f(MyClass $my_inst): string {
13     // Fix me!
14     return $my_inst->alpha();
15}
```

Why did Facebook build the hack typechecker in OCaml?

 Answer

 Request ▾

Follow 52 Comment Share 1 Downvote

...

Promoted by Chartio

Build charts & dashboards in minutes with Chartio.

Connect all your business data in one place. Try Chartio for free and get to new insights in minutes.

[Start now at chartio.com ↗](http://chartio.com)

...

8 Answers



Jon Harrop and Tikhon Jelvis upvoted this



Jean Yang, worked at Facebook

Answered Mar 21, 2014 · Upvoted by Hugo Venturini, works at Facebook and Roy Williams, worked at Facebook



Because [Julien Verlaguet](#)* is French and therefore loves OCaml.

Also because OCaml's strong static typing makes it easy to write correct programs.

OCaml's data types, pattern matching, and higher-order functions also make it really nice for doing the tree traversals involved in building compilers.

* One of Hack's creators (see [Facebook Introduces 'Hack,' the Programming Language of the Future ↗](#)).

Flow

"Using data flow analysis, Flow infers types and tracks data as it moves through your code. You don't need to fully annotate your code before Flow can start to find bugs."

OCaml provides:

- Algebraic data types
- Pattern matching (operate on Abstract syntax trees)
- Statically typed

Rust

- compiler originally written in OCaml (rustboot)
- See Hey! There's OCaml in my Rust!

MirageOS

"is a library operating system that constructs unikernels for secure, high-performance network applications across a variety of cloud computing and mobile platforms."

Unikernels

"Unikernels are specialised, single-address-space machine images constructed by using library operating systems."

DOI:10.1145/2541883.2541895



Article development led by **ACM Queue**
queue.acm.org

What if all the software layers in a virtual appliance were compiled within the same safe, high-level language framework?

BY ANIL MADHAVAPEDDY AND DAVID J. SCOTT

Unikernels: The Rise of the Virtual Library Operating System

Composition in Internet-of-Things using Unikernels

Pradeep Kishore Gowda

Indiana University-Purdue University Indianapolis

Indianapolis, USA

pgowda@iupui.edu

Abstract—This paper describes the concept of unikernel operating systems in the context of Internet of things and cloud computing. First, the growth, challenges and opportunities presented by internet of things and cloud computing, and a review of the factors influencing the development of unikernels as a class of specialised operating systems is provided. We then propose a framework for composing unikernels to provide dynamically configurable systems at build time as well as run time.

Index Terms—unikernels; operating systems; internet of things; service composition; cloud computing.

I. INTRODUCTION

The rest of the paper is organised as follows: Unikernels are described in Section 2, Internet of things is discussed in section

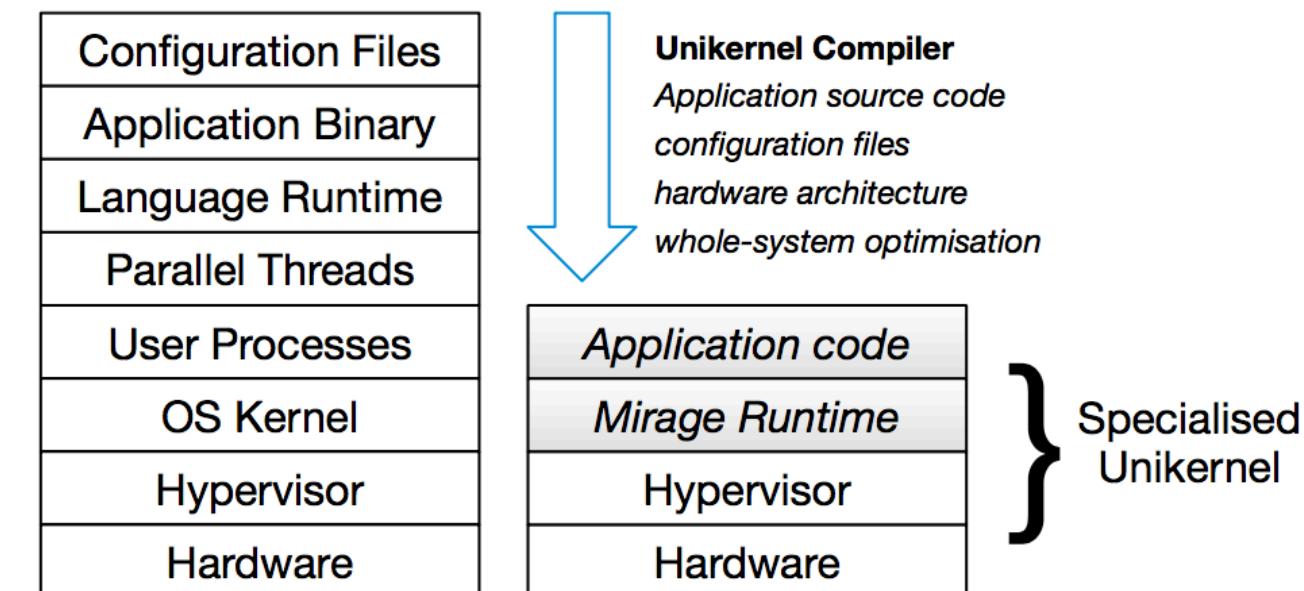


Fig. 1. Compare the software layers in virtual machine appliances vs a unikernel's (MirageOS) stand-alone kernel compilation approach

Unikernel operating systems as network security appliances: a survey

Pradeep Kishore Gowda

Indiana University-Purdue University Indianapolis

Indianapolis, USA

pgowda@iupui.edu

Abstract—This paper describes the concept of unikernel operating systems which have come into existence by the recent developments in cloud-computing and internet of things. First, the growth and challenges faced by specialised operating system demands are explored, and a review of factor influencing the development of unikernels as a class of specialised operating systems is provided. Then, the architecture of present unikernel operating system implementations is outlined, and the methodologies and design-decisions behind each unikernel operating system are explored. Open research issues for the realization of unikernels in the area of network security appliances are also discussed.

Index Terms—unikernels; operating systems; internet of things; cloud computing.

kept there for backward compatibility and more likely inertia [4].

There has been an increased effort to break up operating systems to smaller, manageable pieces to make sharing of service components by guests configurable and auditable, making exposure to risk explicit, and access to the hypervisor is restricted to the least privilege required for each component [5].

Energy efficiency in data centers that provide cloud computing facilities is another active area of research [6] and industry focus [7], that is directly impacted by use of virtual machines that are oversized and duplicate an entire computer infrastructure

NEWS / TECHNOLOGY / GLOBAL / NORTH AMERICA

Docker Buys Unikernel Systems, Plans to Mainstream Virtual Library OSes

21 Jan 2016 6:00am, by [Joab Jackson](#)

Bucklescript

- <https://bucklescript.github.io/>
- **BuckleScript: write JS faster, safer and smaller**
- Created at Bloomberg
- Compiles OCaml to Javascript
- [Playground](#)

What problems does Bucklescript solve?

BuckleScript is mainly designed to solve the problems of large scale JavaScript programming

Type safety

"No verbose type annotation required compared with TypeScript".

"a **sound** type system which means it is guaranteed that there will be no runtime type errors after type checking."

High quality dead code elimination

- Function and module level elimination
- bundle tools like [Google Closure](#)

Fast compilation

You Retweeted

 **Sebastian Nozzi** @sebnozzi · Apr 1
@codemonkeyism Was doing it wrong (S script-exec vs. K compiling).
New, hello-world:
S ~5s
K ~3.7s
TS ~1.8s
Haxe ~0.1s :-D
OCAML ~0.02s o_O

View conversation

S=scala, K=kotlin, TS=typescript.

More efficient code

Small JS output

- In BuckleScript, a Hello world program generates **20 bytes** JS code instead of 50K bytes.

Maintain code structure

- One-to-one mapping between Bucklescript(Reason) -> Javascript output.

Readable javascript output

```
'use strict';
var Pervasives = require("bs-platform/lib/js/pervasives");
var Http      = require("http");

var hostname = "127.0.0.1";

function create_server(http) {
  var server = http.createServer(function (_, resp) {
    resp.statusCode = 200;
    resp.setHeader("Content-Type", "text/plain");
    return resp.end("Hello world\n");
  });
  return server.listen(3000, hostname, function () {
    console.log("Server running at http://" +
    (hostname + ":" + (Pervasives.string_of_int(3000) + "/")));
    return /* () */0;
  });
}

create_server(Http);
```



source

```
"use strict";
var Int_map=require("./int_map.js");
function test() {
    var m = /* Empty */0;
    for(var i = 0; i <= 1000000; ++i){
        m = add(i, i, m);
    }
    for(var j = 0; j <= 1000000; ++j){
        find(j, m);
    }
    return /* () */0;
}
test(/* () */0);
```



source

Why Bucklescript matters for

Javascript platform

ReasonML

Quickstart

Quickstart

```
npm install -g bs-platform  
bsb -init my-first-app -theme basic-reason
```

```
cd my-first-app  
npm run build
```

```
node lib/js/src/demo.js
```

[...jump to Code...]

Syntax

- <https://reasonml.github.io/guide/language/overview>

Code Optimization

ReasonML:

```
let muli a b => a * b;
```

JS:

```
var Caml_int32 = require("stdlib/caml_int32");
var muli = Caml_int32.imul;
```

Code optimization (2)

```
let y = [|1, 2, 3, 4, 5|];
```

```
let twice x => x * 2;
```

```
Array.map Js.log (Array.map twice y);
```

```
y |> Js.Array.filter (fun x => x > 2)
|> Js.Array.filter (fun x => x % 2 == 0)
|> Js.log;
```

Playing with ReasonML

Playground

- <https://reasonml.github.io/try>

The screenshot shows the ReasonML playground interface. At the top, there's a navigation bar with links for Quick Start, Try, Guide, API, Community, Blog, and GitHub. Below the navigation bar are four code editors:

- Reason:** Contains the following code:

```
1 let x = 10;
2 Js.log x;
```
- javascript:** Contains the generated JavaScript code:

```
1 // Generated by BUCKLESCRIPT VERSION 1.0.1
2 'use strict';
3
4
5 console.log(10);
6
7 var x = 10;
8
9 exports.x = x;
10 /* Not a pure module */
```
- ocaml:** Contains the following OCaml code:

```
1 let x = 10
2 let _ = Js.log x
```
- auto-evaluate:** Contains the output of the evaluation:

```
10
(iframe)
```

- [codepan](#)

visual studio code

Where is it used?

- over quarter of the code on [messenger.com](#)

React-Reason

- Reason bindings for ReactJS docs

It integrates deeply with language level features in order to create an expressive, statically typed API, packed into a tiny API surface area.

By binding directly to ReactJS, ReasonReact gives you access to the entire React ecosystem, so that you can adopt it incrementally.

- example project -- <https://github.com/reasonml-community/reason-react-example>

**Thank you
Questions?**