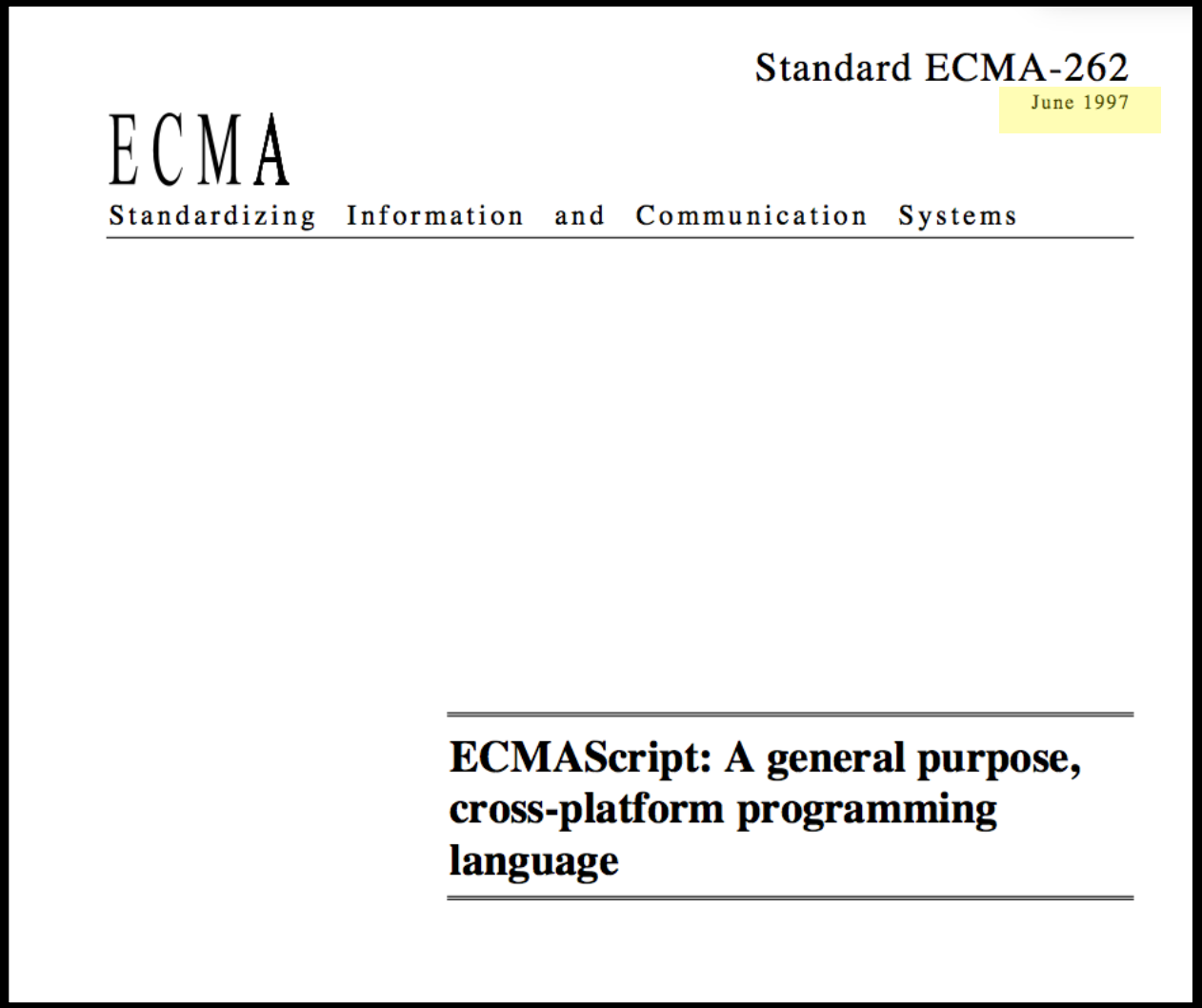
TC39 and ECMAScript
What's in store for the next 20 years?

Allen Wirfs-Brock
May 24, 2017

ECMAScript will be 20 years old next month



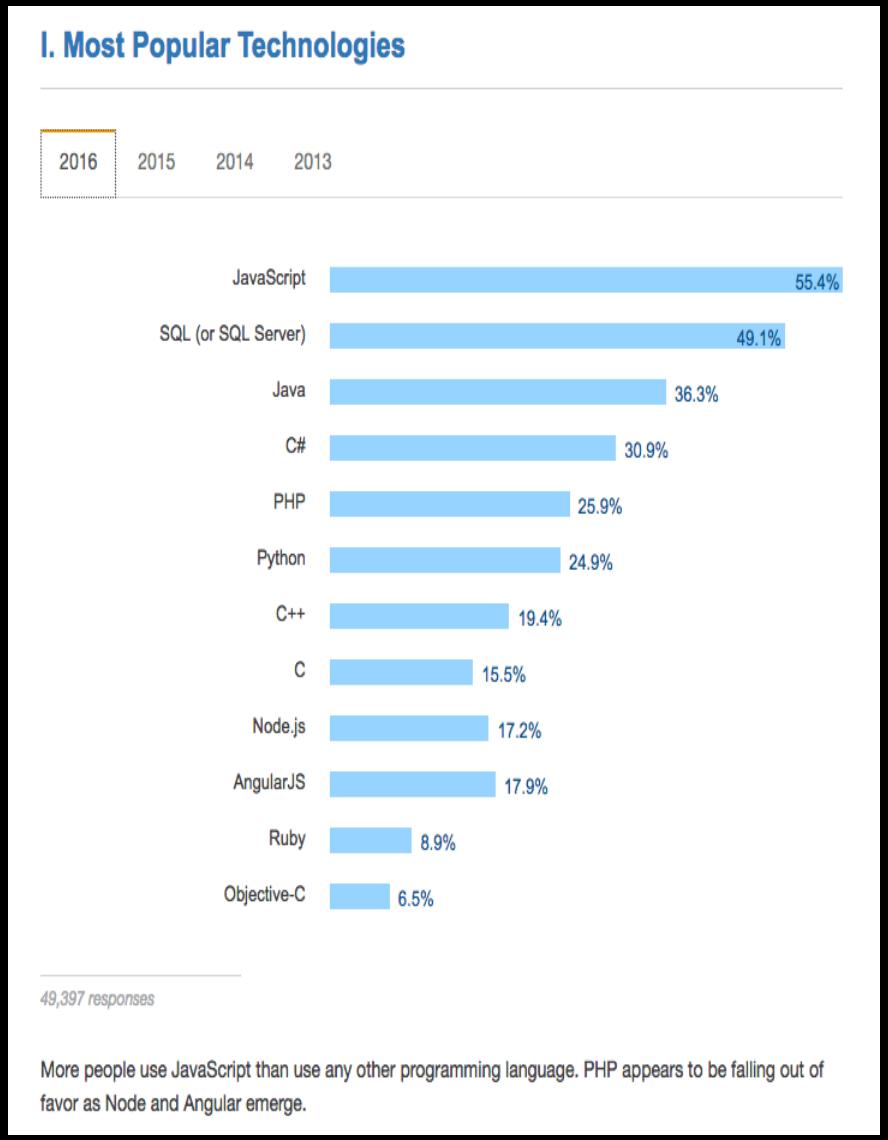
Standard ECMA-262

June 1997

ECMA

Standardizing Information and Communication Systems

ECMAScript: A general purpose, cross-platform programming language



Stack Overflow 2016 Developer Survey
http://stackoverflow.com/research/developer-survey-2016

2016's most popular programming language: JavaScript

1 JavaScript
2 Java
3 PHP
4 Python
5 C#
5 C++
5 Ruby
8 CSS
9 C
10 Objective-C

RedMonk Top 10 Programming Languages January 2016

http://redmonk.com/sogrady/2016/02/19/language-rankings-1-16/

The Floppy Disk

http://www.robert-drummond.com/2015/06/19/in-what-universe-is-javascript-the-new-c/

"Javascript is the new C"

realtime applications, C's real value emerged from that ubiquity: C was the only truly platform-independent programming language. By 1990, you could write a C program and run it on any computer in existence.

Today Javascript has taken over that mantel. C/C++ is no longer universal. *Only Javascript will run on Windows, Linux, OS X, iOS and Android platforms.*

your GPIO I/O ports over the internet

How To Install Windows 10 IoT on a Raspberry Pi

as an extravagance, C was a revelation. Near-assembler fast, bit-level operations, but still an expressive 3GL.

OK, so nothing like JavaScript then?

Wait, that's only the beginning of the story.

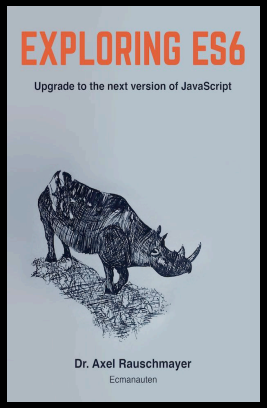
```
        CMP   AX,122    ; compare AX t
        JG    DONE      ; if greater,
        SUB   AX,32     ; subtract 32
DONE:   RET             ; return to ma
SUB32   ENDP            ; procedure en
```

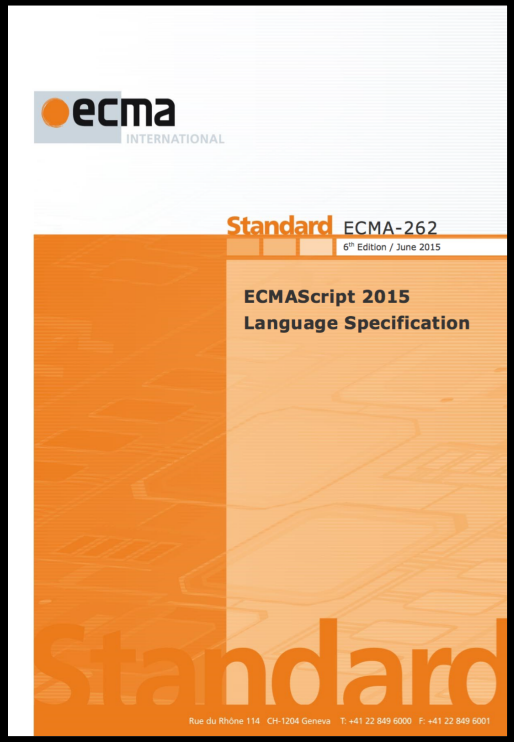
Each Computing Era has had Canonical Programming Languages

Corporate Computing Era – COBOL/Fortran

Personal Computing Era – C/C++ family

JavaScript: The Canonical Language of the Ambient Computing Era





TC39 isn’t just a rubber stamp standards body.

It’s the design team for this era’s canonical programming language.

As JavaScript grows and evolves, so must TC39

Today, TC-39 isn't like either of these

Challenge:
How can TC39 scale to this

Challenge:

Is it at all meaningful to talk about achieving consensus at a 100+ person meeting?

Challenge:

Without a BDFL, how do we continue to evolve JavaScript while preserving a coherent design esthetic

Harmony/ES6 had a plan

http://wiki.ecmascript.org:80/doku.php?id=harmony:harmony

Requirements

1. New features require concrete demonstrations.
2. Keep the language pleasant for casual developers.
3. Preserve the "start small and iteratively prototype" nature of the language.

Goals

1. Be a better language for writing:
   I. complex applications;
   II. libraries (possibly including the DOM) shared by those applications;
   III. code generators targeting the new edition.
2. Switch to a testable specification, ideally a definitional interpreter hosted mostly in ES5.
3. Improve interoperation, adopting *de facto* standards where possible.
4. Keep versioning as simple and linear as possible.
5. Support a statically verifiable, object-capability secure subset.

Means

1. Minimize the additional semantic state needed beyond ES5.
2. Provide syntactic conveniences for:
   I. good abstraction patterns;
   II. high integrity patterns;
   III. defined by desugaring into kernel semantics.
3. Remove (via opt-in versioning or pragmas) confusing or troublesome constructs.
   I. Harmony builds on ES5 strict mode to avoid too many modes.
4. Support virtualizability, allowing for host object emulation.

First version Aug 2009
Updated as need

This snapshot July 2011

A Possible Exemplar: WG21

https://isocpp.org/std/the-committee

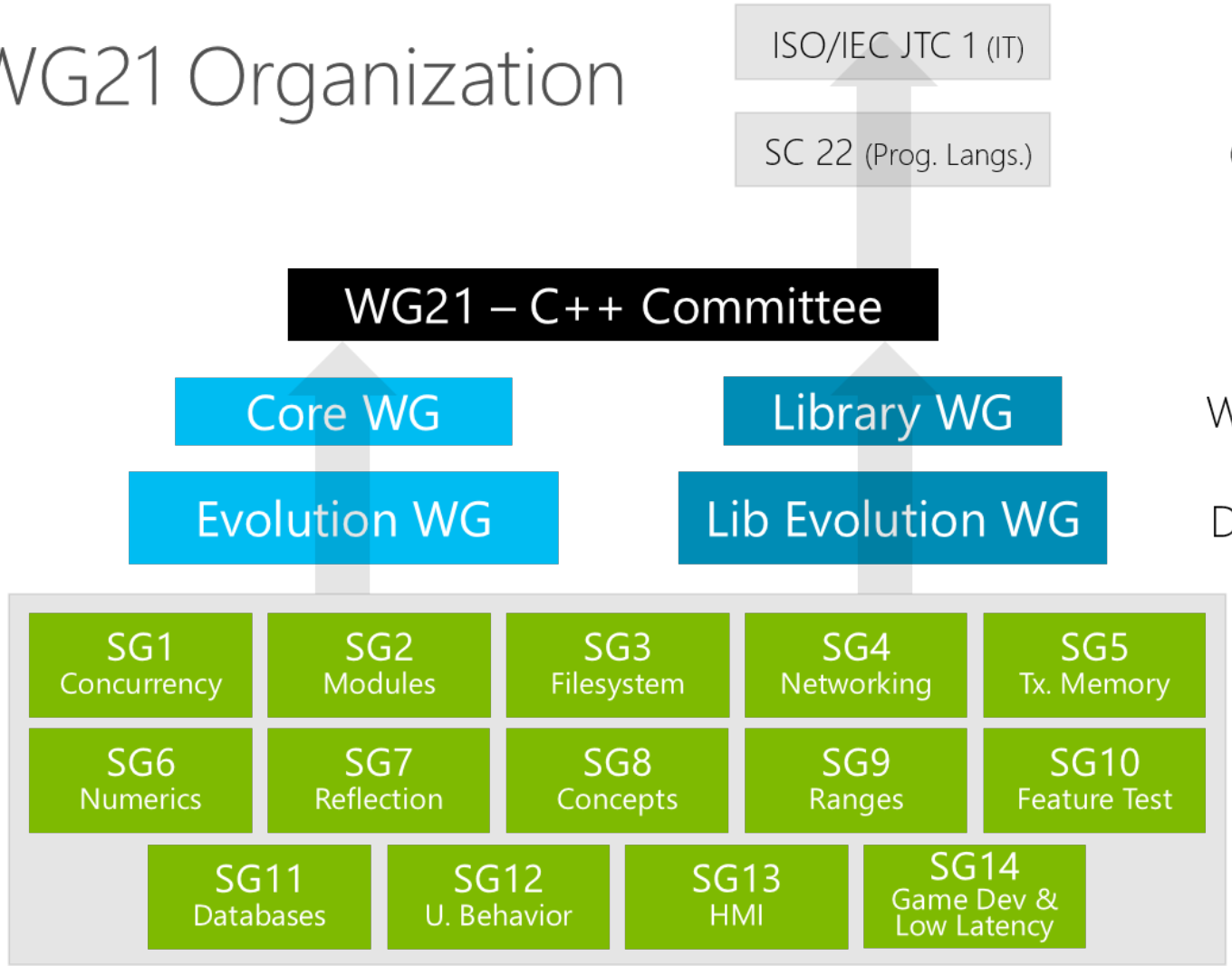
The C++ committee holds two or three full week-long face-to-face meetings a year. One meeting a year is traditionally held outside the continental United States – often in Europe, but periodically in Canada or Hawaii, or occasionally in the Caribbean, Japan, or Australia.

Typical attendance ranges from 90 to 100 people. These are five- or six-day meetings (Mon-Fri or Mon-Sat), and begin and end with everyone in the same room for a plenary session: On Monday morning, we meet together to organize work for the week, and at the end of the week we meet to consider change recommendations ready to be brought before the whole committee for approval polls. The rest of the time is spent in smaller subgroups where most of the technical discussions occur.

Completed
C++17

Kona, HI, USA
(2017)

WG21 Organization



(F)DIS Approval

CD & PDTS Approval

Internal Approval

Wording & Consistency

Design & Target (IS/TS)

Domain Specific Investigation & Development

Part 2

Sometimes it’s a good idea to reexamine fundamental assumptions and early design decisions

ECMA-262 Edition 1
Clause 4, Paragraph 1

ECMA-262, Edition 1 (1997) said:

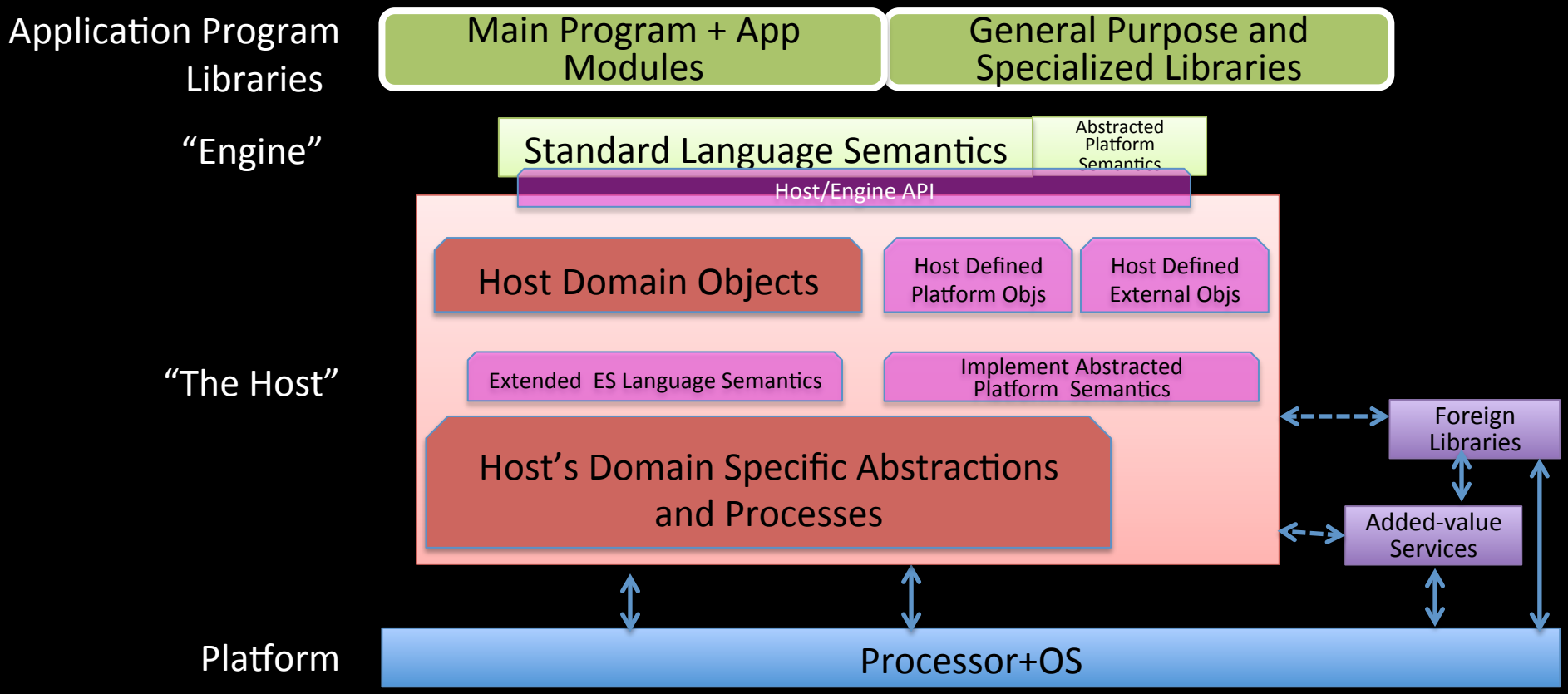
EMCAScript is an object-oriented programming language for performing computations and manipulating computational objects within a host environment. ECMAScript as defined here is not intended to be computationally self-sufficient; indeed, there are no provisions in this specification for input of external data or output of computed results. Instead, it is expected that the computational environment of an ECMAScript program will provide not only the objects and other facilities described in this specification but also certain environment-specific *host* objects, whose description and behavior are beyond the scope of this specification except to indicate that they may provide certain properties that can be accessed and certain functions that can be called from an ECMAScript program.

Draft ECMA-262 2018 says:

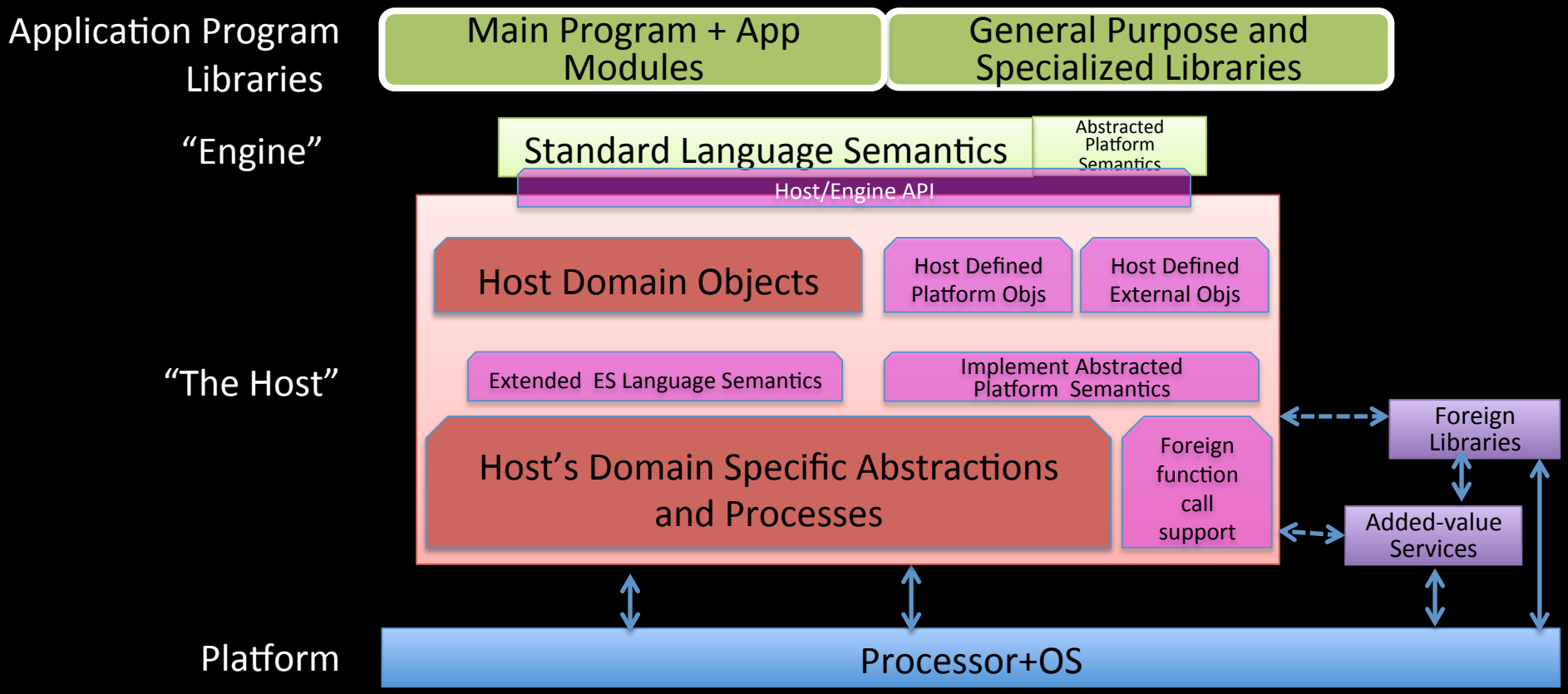
ECMAScript is an object-oriented programming language for performing computations and manipulating computational objects within a host environment. ECMAScript as defined here is not intended to be computationally self-sufficient; indeed, there are no provisions in this specification for input of external data or output of computed results. Instead, it is expected that the computational environment of an ECMAScript program will provide not only the objects and other facilities described in this specification but also certain environment-specific ~~*host*~~ objects, whose description and behavio~~u~~r are beyond the scope of this specification except to indicate that they may provide certain properties that can be accessed and certain functions that can be called from an ECMAScript program.

Why Does ECMAScript Need a “Host”?

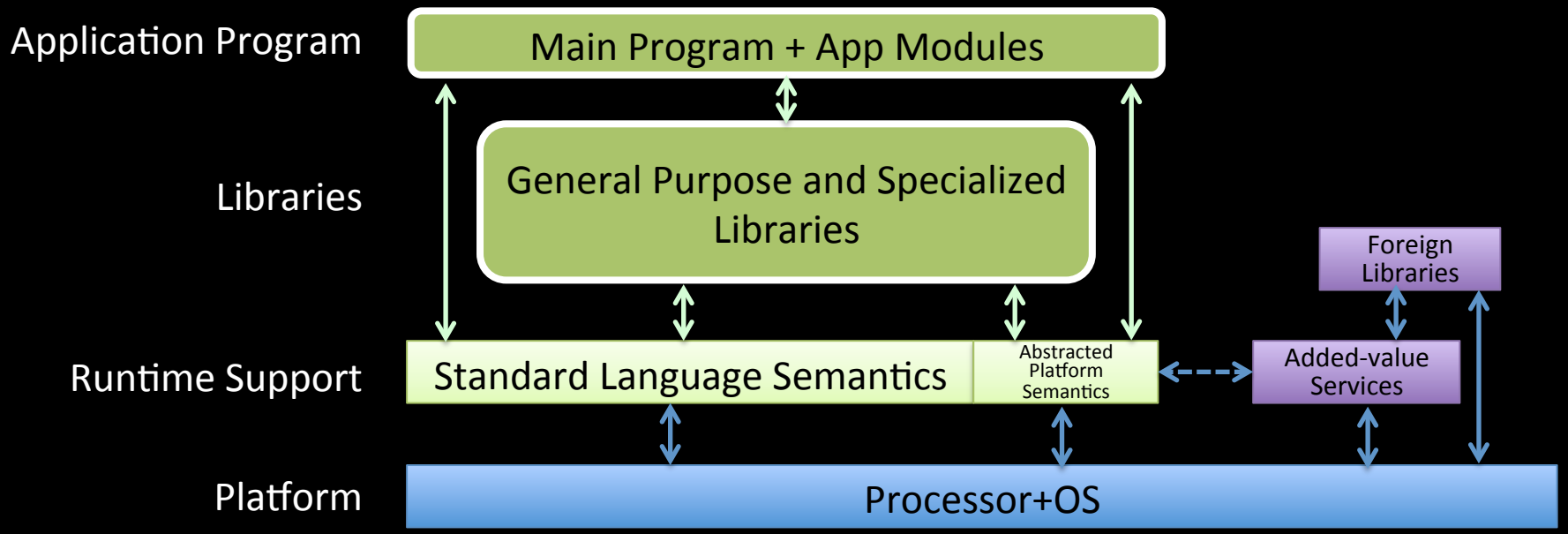
Hosted ECMAScript Platform Integration



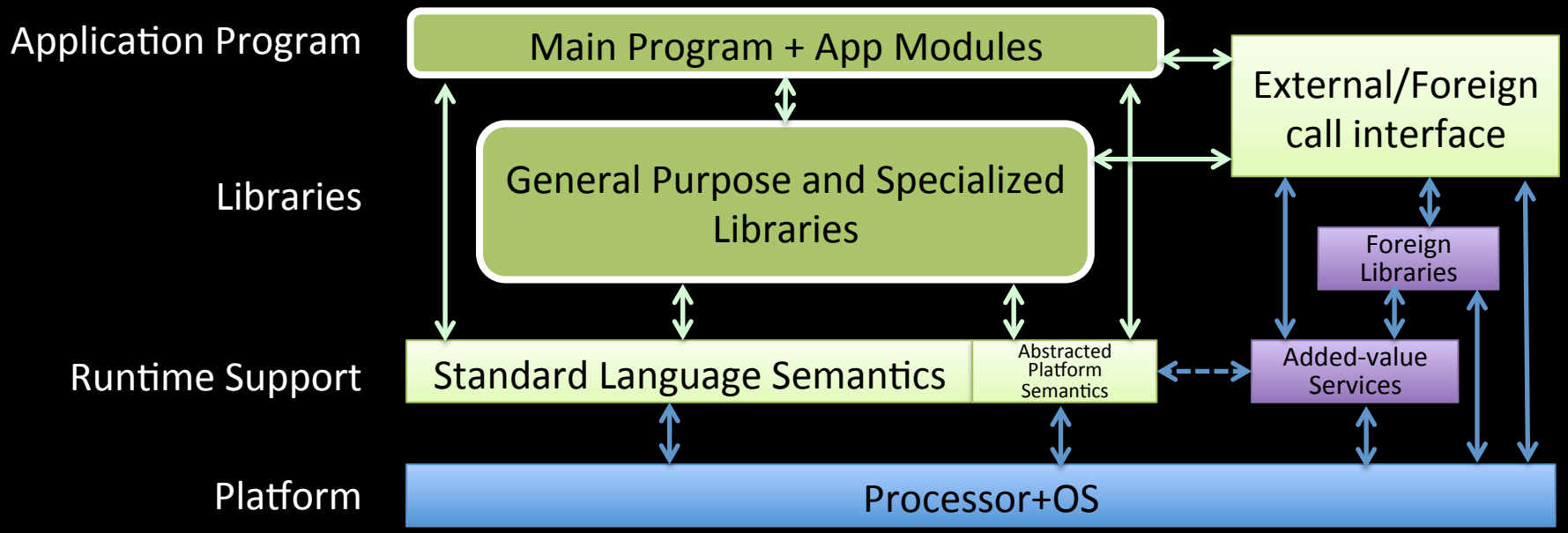
Hosted ECMAScript Platform Integration w/ Host FFI



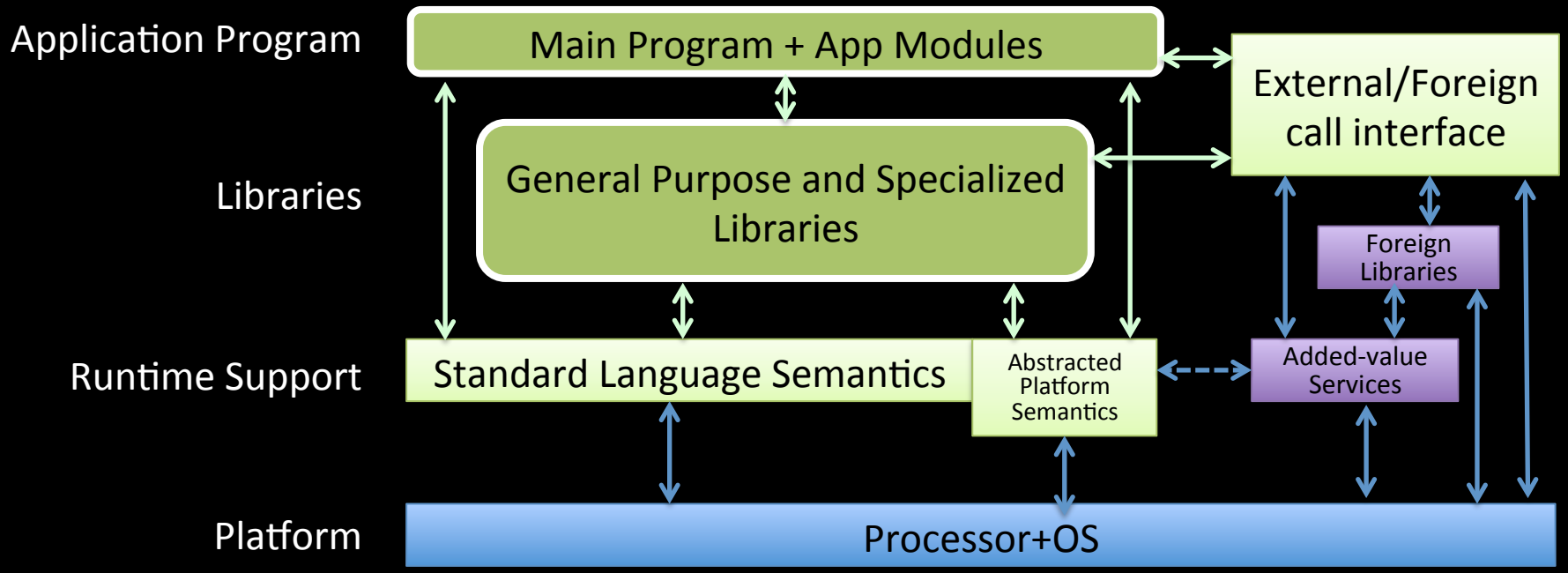
Conventional Programming Language Platform Integration



Conventional Programming Language Platform Integration w/ FFI



Hostless ECMAScript Platform Integration?



Could we build this today? Not quite 😕

- Things that are missing and currently defined by hosts
  - Incomplete set of abstracted platform semantics; not currently specified in the ECMAScript standards
    - Eventing
    - Worker/Agent/Threads and control API
    - Realms
    - Module loading reflection
  - A standard Foreign Function Interface specification
    - EG, define a FFI module syntax