javascripts

in the javascripts

ffconf 2014

andy wingo


BIG KID CIRCUS
BIG KID CIRCUS
ERF
ES6

the
es6
circus
is
coming
to
town

`es-discuss` clownshoes

C++ knife-jugglers

JavaScript acrobats

building es.next in es.now

Hark, an agenda:

- Why?
- How: JavaScriptCore
- How: SpiderMonkey
- How: V8

why
implement
js in
js?

js is
faster
than
c++

js is faster than c++

JS can optimize in ways that C++ can't

- dynamic inlining
- inline allocation (and possibly scalar replacement)
- inline hard-wiring of user object shapes (slot offsets, getters)

js is faster than c++

No JS/C++ transition cost

Especially important for callbacks (e.g. `forEach`)

js is faster than c++

JavaScriptCore's Oliver Hunt, January 2014:

"The initial proof of concept is `Array.prototype.every`, this shows a 65% performance improvement, and that improvement is significantly hurt by our poor optimisation of `op_in`."

js
matches
js
semantics
better

Proxies, accessors, order of effects, has-property versus get-property, user-implemented iteration protocol, exceptions, catch

Terse:

```
for (var x of y) z(x);
```

js more secure than c++

GC-related bugs approximately impossible

- SM, V8; JSC immune

No C++ knife-throwing work-related accidents

- integer overflow, use-after-free, etc

Cross-`iframe` leakage concerns lessened

choosy
hackers
choose
js

Goal: As much in JS as possible

For speed, for security, for maintainability

How?

simplest model: javascriptcore

“Methods can be implemented in JS”

example

Source/JavaScriptCore/builtins/Array.prototype.js

```
function foo() {
  return 'ahoy ffconf';
}
```

Source/JavaScriptCore/runtime/ArrayPrototype.cpp

```
foo arrayProtoFuncFoo DontEnum|Function 0
```

weird js: jsc edition

Function source compiled separately

Access to globals forbidden in general

Initial values of globals accessible via @ prefix, e.g. `@Object`

Add `@call` and `@apply`

`http://svn.webkit.org/repository/webkit/trunk@163195`

more complicated: spider monkey

“Self-hosted JS” files concatenated and evaluated – more normal model

C++ binds functions by name to prototype properties

feature: es.next ‘pipelines’

Old SpiderMonkey:

```
(x*2 for (x in [0,1,2].keys()))
```

Erstwhile ES6:

```
(for (x of [0,1,2].keys()) x*2)
```

Maybe ES7:

```
[0,1,2].keys().map(x=>x*2)
```

Ideally on `IteratorPrototype`, but let’s hack it

example

js/src/builtin/Iterator.js

```
function* IteratorMap(f) {
  for (var x of this) yield f(x);
}
```

example

No `function*` at boot-time :(

But, ES6 object literals

```
function IteratorMap(f) {
  var iter = this[std_iterator]();
  return {
    next(val) {
      var result = iter.next(val)
      return result.done ? result : {
        value: callFunction(f, iter,
                            result.value),
        done: false
      };
    },
    [std_iterator]: IteratorIdentity,
  }
}
```

example

Link to C++ files; grep for surrounding identifiers, make similar modifications (e.g. in `jsiter.cpp`)

```
js> for (var x of [1,2,3].keys().map(x=>x*2))
      print(x)
0
2
4
```

nerf
the
web
forward

nerf
the
web
forward

Your search - "nerf the web forward" - did not match any documents.

nerf the web forward

(like, nerf is like a more resilient polystyrene foam)

nerf
the
web
forward

(the more joke explanation slides, the more amusing the joke, right?)

nerf
the
web
forward

(right?)

caveats

`@@iterator` called before or after first `next()`?

Prototype chain of the result of `map()`?

Should final `result.value` be mapped?

`%IteratorPrototype%`

No spec; spec wonkiness

`throw()`?

`next()` applied to different object?

v8

Story time!

languages are like operating systems

Visit a page : Install an app

Visit `about:blank` : Boot OS

Weird self-hosted JS part of OS, not app

genesis

In the beginning, there was the empty function

and the `Object` function

and its `prototype` property

genesis

And Goog looked upon it and saw that it was good

genesis

Then the strict mode function “maps” (hidden classes)

Then the first global object

Then `Array`, `Number`, `Boolean`, `String`, `Symbol`, `Date`, `RegExp`, `JSON`, `ArrayBuffer`, the `TypedArrays`, `Map`, `Set`, iterator result shapes, `WeakMap`, `WeakSet`, arguments object shapes, …

genesis

And Goog looked upon them and saw that they were good

genesis

And Goog looked upon them and saw that they were good

But FFS it’s a lot of C++, innit?

how 2 js

Problem: Need to define helpers in JS, but they shouldn’t be in the user’s scope

Solution: Second global object for self-hosted JS to play in; natives mutate to produce a more beautiful global

`builtins`,

`globals`

Global: A global object, corresponding to a user-facing script-level scope

`builtins`: The global object current when self-hosted JS is being defined

In `builtins`, user-facing global bound to `global`

Somewhat confusingly, in V8, “self-hosted JS facilities” are called “natives”

on the seventh day

So, “natives”. That’s JavaScript y’all!

example

src/generator.js

```
function* GeneratorObjectMap(f) {
  for (var x of this) yield f(x);
}
```

weird js, v8 edition

Verbs

- `%` prefix for low-level C++ runtime functions (`--allow-natives-syntax`)
- `%_` prefix for magical “inline” runtime functions (`%_CallFunction`, `%_IsSmi`)
- macros (`TO_UINT32`, `IS_NUMBER`)

weird js, v8 edition

Nouns too

- `global`
- `InternalArray` (to allow builtins to use `.push()` without worrying about user pollution)

Suggested reading order

- `runtime.js`
- `v8natives.js`
- `array.js`

snapshots

Lots of work amirite?

Optimization: Serialize heap of new-born world

Load fresh heap from disk to “boot”

Necessary in context of Chrome’s multi-process model

note: the dom is something else

- “Blink-in-JS”
- Kentaro Haro: DOM binding overhead is 5-15% in real web
- DOM objects live in a 1-to-N relationship to V8 globals
- Search for “Hardening security of content scripts”

but seriously

- Strict spec reading
- Strict spec translation (optimize later if ever)
- Tests (especially proxies, getters, order of operations)
- Patch submission
- Feature flags (in v8)

tx

nerf the web forward!

wingo@igalia.com

http://wingolog.org/

.

big kid circus, by ray forster: https://www.flickr.com/photos/94418464@N08/8686092191