Notes - 3/28

Monday, May 02, 2011 7:59 AM

Will we degrade or throw?

- Safely infer
- Host specific method:
  - Dummy on other browsers (or instanceof)
- Does throwing behavior need to be consistent across browsers?
  - Dial for higher-perf mode
- One of the big things we gain is freedom from stupid errors
  - When it really starts hurting performance
  - Even runtime checked
- Compiler's responsibility to find errors, runtime's responsibility to

How much work should the runtime be doing to enforce typing?

```
function foo(int x) {
  int _x = x | 0;
  _x = (_x+1) | 0;
}
```

But if you call foo(1e100) it just silently clamps and makes it 0.

Maybe instead:

```
foo(int x) {
  Var _t = +x;
  Var _x = _t|0;
  If(_x !== _t) throw TypeError()

}
```

Use NaNs to get into the throw clause

Should we be throwing on

What does this mean for the static type system?

-

Anders: Implicit conversions never lose information. Explicit conversions presumably could lose information.

This has to happen at every public method boundary.

Arrays?

- Can we use typed arrays here?

For classes, we are strict, but give a constructor to get into the conforming system.

- Could do the same for arrays, give them a constructor for taking a normal array and making it a typed array

Front-end initializes unassigned variables so that perf box can find type assignments.

Perf box will require define before use – and the definite assignment must define the type.

- Then it's up to strada to generate the right thing

Same applies to constructor:

- When loading

Locals and members through initialization

What needs to be annotated
- Internal methods and privates don't have the guards
- Seems to only be necessary for privates across separate compilation units of the same module
    - Perhaps create a fake caller in the same compilation unit

Private, internal, public:
- Previously, just internal and public
- Concern that the checking is going to look like it is slow, even if it isn't
- Introduce a category of

No PBInt, just values (0).

IE10 supports a JavaScript extension called enforceType. Other browsers which do not support this extension will not throw.
- There will be browser differences
- As long as IE10 doesn't throw, there will not be differences – when IE10 does throw, other browsers may not throw
- This code uses this extension,

Plan is at load time to create static entry points, and to send strongly typed calls there

Once you have adopted a particular representation, you have to protect that representation.
- Two choices: throw, or flip a big red switch

Mads' alternative:
- Can't get rid of the throwing, but can potentially get rid of

Could have a mode that says "complete consistency":

Guarding against safety is important, but going further probably isn't.

```
Internal foo(ing x, Rect r) {
…

}


<<foo>>(int x, Rect r) {
  // strongly typed
}

Foo$$(x,r) {
  guard(x,int);
  guard(r, Rect);
  return <<foo>>(x,r);
}  // or alternatively, just throw


Strada: (1)
Move(int dx, int dy) {
  X += dx;
  Y += dy;
}

Generated: (2)
Move(dx, dy) {
  _x += dx;
  _y += dy;
}
```

IE10 bifurcates generated code into:
<<move>> (3) and _move (4)

Chrome puts (2) in the prototype, we put (4) in the prototype

This looks like a good plan.

How does someone get here:

Strada.enforce(Point) -> Strada.oprtimize(Point)

```
function c() {
  …
}
```

Perhaps there is a member that just calls everything?

Leverage definite assignments to get type assignments.

Need a private convention that is short and evocative and recognizable

Why to distinguish private and internal?
- Don't need to – no private

How are we going to enforce a class is a class?
1. Chrome never throws, object just goes through
2. Chrome checks prototype using instanceof
3. GUID field

(2) sounds good.

```
Public foo(Point p) {
// need a this check too…
If(!(p instanceof Point)) throw …
}
```

We replace this with a stronger check in IE10.

It's possible that for internal calls to public methods, we want to skip the guards

If we need to put Chrome on better performance footing, we could generate two entrypoints for each function.
- Worried about code bloat and code complexity

Dials:
- Optimize for verification or optimize for speed
- Steve favors one switch:
  - Vanilla code, non optimizied
  - Optimized

From: Luke Hoban
Sent: Monday, March 28, 2011 9:53 AM
To: Strada Design Team
Subject: Strada Design Meeting - 3/28

Here are the agenda items on our backlog that we can discuss today unless there are other pressing issues that came out of last week's meeting:
- Any remaining review of metadata encoding (PBCtor)
- Any remaining discussion on throwing vs. degrading perf on violations of metadata
- Array types
- Function types
- Optional parameters
- Varargs
- Typeswitch
- Typing of ES5 builtins (Strada core library)

Also – if there is interest, I can share some takeaways from last week's TC39 meeting related to: private names, classes, modules and async.

Luke