Minutes of the: Ecma TC39-TG1
held in: Phone conference
on: 28th November 2006

Attendees

- Graydon Hoare, Mozilla Foundation
- Cormac Flanagan, UC Santa Cruz
- Jeff Dyer, Adobe Systems
- Francis Cheng, Adobe Systems
- Dan Smith, Adobe Systems
- Dave Herman, Northastern University
- Brendan Eich, Mozilla Foundation
- Pratap Lakshman, Microsoft

Agenda

Dave and Cormac's type system questions:

- is `this` only ever a class type? interface type?
  - [jd] class instance inside of an instance method, global object type in global code, object type in unbound functions
- are there places where `this` can't be mentioned?
  - [jd] inside of an instance method, global code, the body of an unbound function (non-method function definition or function expression)
    - [dh] this answer doesn't make sense to me..?
      - [jd] take two: `this` can only be mentioned inside of an instance method, global code, the body of an unbound function (i.e. non-method function definition or function expression). This means that the only place that `this` cannot be used in a static initialiser of a class or a static method. (note: AS3 also disallows `this` at the top level of a package)
- what are restrictions on `SimplePattern` expression? where are they imposed?
  - [jd] `SimplePattern` is either a PostfixExpression or an Identifier depending on where it occurs syntactically. E.g. in a parameter context it is an Identifier, on the left side of an assignment it is a PostfixExpression
    - [dh] from meeting: in non-binding context: identifier, dot, bracket, or call (`SimplePattern`); in binding context: identifier only (`IdentifierPattern`)
- what is `PatternExpr`?
  - [jd] fuzz. ignore it
- why don't `IdentifierPattern` have a `TYPE option`?
  - is it this: `let ([''dave'', x:int, b:Boolean] = [''dave'', 54, true]) { ... }`
  - or this: `let ([''dave'', x, b] : [String, int, Boolean] = [''dave'', 54, true]) { ... }`
    - [jd] it is the latter: `let ([''dave'', x, b] : [String, int, Boolean] = [''dave'', 54, true]) { ... }`
    - [jd] patterns can be be typed, which results in a TypedPattern

            - [dh] the former version seems more convenient, no? we can reconstruct the type of the entire pattern by recursively propagating the inner type annotations through the structure of the pattern.. there might be ambiguities but I think not; for literals it doesn't matter what the type is, for unannotated variables it's `*`, and otherwise it's whatever the annotation is. this way the programmer could put the annotations directly with the bindings. seems especially appropriate for the case of multiple-value return
                - [jd] the problem is that in a non-binding (assignment) context we don't know if we are seeing an object literal or a pattern, so it is convenient if their syntaxes are the same. This is an implementation issues which could probably be overcome with a heavy (language design) hand, but not without allowing hard to read programs. Let's discuss...
            - [dh] brendan: we've been through this before, the ambiguity introduced in object patterns is confusing, so the second version keeps things unambiguous

- what is a `Ref` expression?
    - I think it's for variable and field dereference, right?
        - [jd] it has been replace by ObjectRef and LexicalRef
- what is the `NULOP Empty`?
    - [jd] fuzz
- bug in `parser.sml`? – `type foo` creates a `TypeExpr`, not a `UnaryExpression` with `UNOP type`?
    - [jd] NAB. `UnaryExpression` takes an `EXPR`, the operand of `type` is a `TYPE_EXPR`. So, we need a different constructor. We could have named it `UnaryTypeExpr`, to mirror what we do with `BinaryTypeExpr`, but there is only one of them so I named it what it is.
- what is the definition of a statically known namespace?
    - is it the following?
        - when you see a qualified name `e1::x` or `e1::e2`
        - the expression `e1` is a simple `LexicalRef` with an unqualified identifier `n`
        - the unqualified identifier `n` is in scope bound to a namespace
        - the namespace was declared via `namespace n;` or `namespace n = ``...'';` or is a built-in namespace like `intrinsic`, `public`, `private`, etc.
    - [dh] a namespace reference is statically resolvable if:
        - it is unqualified (since all open namespaces *must* be statically known)
        - or it is qualified by a statically resolvable namespace;
        - and it refers to a namespace that was declared via `namespace n;` or `namespace n = ``...'';` or is a built-in namespace like `intrinsic`, `public`, `private`, etc.
    - so is it impossible to declare `namespace m::n = ...;`?
        - [fc] Jeff: This is permissible if m::n is a static identifier.
- what is the definition of a statically resolvable name?
    - all open namespaces are statically known?
        - [jd] yes
    - and name can be statically resolved in those open namespaces
        - [jd] yes
- list of current open namespaces
    - does order matter?
        - [jd] order of nesting matters
    - what about "parallel" open namespaces?
        - [jd] namespaces opened in the same scope are parallel and references that resolve to different defintions in both are ambiguous
    - does it need to be a list of lists?
        - [jd] a list of sets would suffice

    - [dh] so it's a list of lists (or of sets, but lists are simpler), where the outer dimension is ordered by block nesting, and the inner dimension is unordered; i.e., all `use namespace` pragmas within the same block are considered siblings
- in `IDENT_EXPR`, we split out the cases of static names and computed names:

```
and IDENT_EXPR =
      QualifiedIdentifier of { qual : EXPR,
                               ident : USTRING }
    | QualifiedExpression of { qual : EXPR,
                               expr : EXPR }
```

    - should we also split out the cases of static namespaces and dynamic namespaces?
    - or is it not a syntactic criterion? i.e., there must be an ML function `isStaticNamespace`?
        - [jd] the value of qualifiers is not generally known until runtime, so in AS3 we treat all qualifier expressions are dynamic. It might be possible to improve on this in ES4. E.g. `private::x` could be an considered statically qualified. So could `q::x` when `q` is known at compile-time (resolved during the definition phase)
            - [dh] I thought about this some more; the type checker needs to distinguish static namespaces from dynamic namespaces, but since it requires knowledge of the lexical environment to determine whether a namespace is static, we probably should just leave this as an `EXPR` and let the type checker determine whether the namespace is static via `isStaticNamespace` or some such

Dave and Cormac's ML question(s):

Does anyone have a better solution for dealing with uncaught exceptions? (we're currently inserting debugging info for catch-all match clauses)

- [gh] found a special mode in SMLNJ that may address this. Will forward info to the list about it.

Notes

Assignment example:

```
var obj = { w: 10 }
obj.w // 10
function f() { return obj }
[f().w] = [42]
obj.w // 42
```

- Notes for Dave and Cormac's questions are inline (above).

- [jd] Need to pin down the AST.

- [be] There are a lot of unassigned tasks, I'll go in and take some of them, maybe we should all take a look.
    - [dh] Maybe better to do this at the next f2f.

- [be] About face to face meetings, let's move to three-day meetings. Tentative schedule:
    - Dec 13, 14, 15 at Adobe San Francisco
    - Jan 24, 25, 26 at Mozilla
    - Feb 21, 22, 23 at Adobe San Francisco

    - Mar 21, 22, 23 at Microsoft Redmond

- [jd] We should discuss structure of the spec, both overall architecture and how to integrate the ML code. [dh] Would it help to have a tool that inlined ML code directly into the wiki? [gh] The wiki source is all plain text, so the easiest way would be to use a Perl script, but I don't know about remote access to the wiki source. [dh] It would also be helpful to have an automated system where the ML source code repository is made available over http somehow. [gh] there are a couple different ways to do that.

- [be] About the overall structure, will work with Francis and maybe Jeff to come up with some options. Will look at other language specs, keeping in mind the structure of the earlier Ecmascript specs. We could perhaps have a mapping of ES3 to ES4 sections. Likes the Scheme approach.

- [jd] So my tentative order of priority for upcoming work is pragmas, definitions, then statements. Does anyone prefer a different order? [gh] and [dh] both think statements should be highest priority.