Minutes of the: Ecma TC39-TG1

held in: Phone conference

on: 27th March 2007

Attendees

- Jeff Dyer, Adobe Systems
- Lars Hansen, unaffiliated

Agenda

- T~

Discussion

T~

- Should we have a shorthand for the union type `(T,undefined)` spelled `T~`?
- We posit that this kind of type exists in builtins and host objects
- The `*` type works for when `T` is `Object`, but not when `T` is a more specific type
- In ES, `undefined` is used to express the idea of "no value provided" in contrast to `null`'s meaning of `no value`
- The distinction is subtle, but already exists in ES3 and so we should support it with convenient syntax
- ACTION: Lars to review builtins to see how common the union type with `undefined` really is
- Attendees agreed that if the shorthand has general use, even if for compatibility with builtins and host objects, then we should support it

Instances in the builtins where this might be useful

This list should not be considered definitive, both because details of some of the methods affect whether the `(T,undefined)` union is actually applicable, and because there may be some I've missed.

```
Array.prototype.join
Array.prototype.slice
Date.UTC (intrinsic static)
Date.prototype.setMilliseconds, setSeconds, ... (maybe)
{Number,int,uint,double,decimal}.intrinsic::toString (probably)
Number.prototype.toExponential, toFixed, toPrecision
Object.prototype.propertyIsEnumerable
RegExp constructor
Name constructor (though this has disappeared again?)
```

There might be a few cases in the DOM, but not many.

IMO the great value of the union type in this situation is that it allows a precise type characterization and out-of-band values ("I want a double here, but it's an optional argument and I want to be able to tell apart NaN and 'no argument passed'"). This type characterization is also backwards compatible to a large extent; a precisely typed builtin library will continue to work with most correct programs that declare `use strict`.

— Lars T Hansen *2007/04/10 06:31*