Notes - 3/1

Friday, March 1, 2013 9:37 AM

[In 41/4541 at 10:00 today.]

Agenda:

- Validating that these are cut for 1.0:
    - Modules implementing interfaces
        - CUT
    - Self-types
        - CUT
    - Strongly-typed this in functions
        - Keep on backlog, but not ready
        - jQuery use case
        - function foo(this: jQuery, args: { }) {}
- Spec questions on 0.9 designs:
    - Class side inheritance (super and this)
        - __extends codegen
        - Super in static: No, you can just say Point.foo
        - This in static: Allowed, with type that is the constructor object type minus all construct signatures.

```
interface StaticFoo {
   //new (): Foo;
   bar(): StaticFoo;
}

class Foo {
        static bar(): StaticFoo {
                return this;
        }
        static baz() {
                console.log("Goodbye")
        }
}

//class Foo2 extends Foo {
// static baz() {
// console.log("Hello")
// }
//}

//Foo2.bar() // "Hello"
```

-
    - Void as a type argument
        - Void can now end up being a type of a property/param when you derive from Foo<void>
        - But we don't let you write "foo: void" as a property directly
        - Spec currently says 'void' allowed everywhere
        - Could have tried to constrain for locals
            - But we won't
    - Tuple types

        ```
        // if bar is a literal number or string and the member named ToString(bar) exists on the type of foo -> return type of foo.bar
        // else if bar is assignable to number and foo has a number indexer -> return type of number indexer
        // else if bar is assignable to string and foo has a string indexer -> return type of string indexer
        ```
    - Should ‘in’ operator allow number on LHS?
        - "i in val" allows i to be number, string and any
        - Because this matches the typing of !!val[i]
    - Best common type
        - According to language, never an error, and 'any' if no common type
        - If 'inferAny' flag is on, that error is triggered here
    - Do enums have a number indexer?
        - Yes, see onenote
    - Static initialization at end of class body?
        - Move all the declarative initialization to the beginning, in order
    - “-out” for external modules (and separating “–outdir”)
        - Send out mail with what we think this looks like early next week
- Other potentially open items for 1.0:
    - Module import syntax
    - Let/const/__proto__
        - Assumption: Not in 1.0
    - Thin arrow
    - Recursive type check improvements
    - Extending 'declare var'
    - Remaining class alignment deltas:
        - Class expressions
        - Extends expressions
        - “extends null”

Luke