Notes - 7/15: SENT

Thursday, July 21, 2011 8:56 PM

- Update on implementation from Steve

Memory for types is an issue:

- Aggressively cache
- Don't do a merge more than once
- **Swiss army type is very nice**
- **Normalizes fields into properties with types**
- **Module/class merging is clean**

What can merge?

```
Module Outer
{
  Module foo
  {
    Public x = 1;
  }
  Public foo(s : string) : string { }
}
```

Question: Should we restrict modules to being immediately after the definition? Maybe
Question: Should classes be partial? No.

Partial:

- **Modules: Imperative, fully partial**

```
if(!this.Outer) this.Outer = {}'
(function(Outer) {
  Outer.foo = foo;

})(Outer);
```

Modules we merge, everything else we smash.

- Notable exception, within a lexical scope, you can extend a class or function definition with a module

Specialized array interface

- **Marker interface IElement**

Great to have a manifest file that specifies the base types

Inheritance is just merging

Dogfoodable drop:

- Hosted in visual studio:
  - Colorization
  - Error squiggles
  - Not intellisense yet (but for CTP)
  - Steve: August language services month
- Compiler:
  - Caught up to design by end of July
  - JSHost for batch compilation
  - Separate compilation
  - Up to date language
- Web front end:
  - Polished and usable
- Spec that is up to date
  -
- Supported compilation models:
  - (1) design-time,
  - (2) server-side and/or

        - What about multiple
    - (3) client-side
    - (4)

Note: We'll start 30 minutes late today at **10:30** due to a conflict.

Agenda:

- Update on implementation from Steve
-
- Implementation blocking issues on merge types?
- Path to dogfoodable drop and CTP
    - Compiler deliverable
    - IDE deliverables prioritization
    - Web-based client deliverable
    - Documentation/spec deliverables
    - Supported compilation models: (1) design-time, (2) server-side and/or (3) client-side
    -
- Backlog:
    - A : B vs. A | B
    - Promises typing
    - Top level scope?
    - Uninitialized variables/inference/defaults
    - O[x] with typed objects
    - Enums
    - Global module
    - Type compatibility rules
    - Foreach
    - Class-parameters as privates and inheritance
    - Function type compatibility (omit parameters, more parameters, varargs, etc.)
    - Any -> number (what gets emitted?)
- Anything else?