Notes - 4/11: SENT

Monday, May 02, 2011 7:59 AM

Anders' thinking about:
- Modules
- Classes
- Interfaces
- Type inference

Fundamental concern: Typing in current system could be viewed as too much work to be worth it.

Type inference spectrum:
- C#
- Scala
- F#

Scala lands in a good spot – return types and members inferred, parameter types inferred.

What is our mindset for existing JavaScript concept:

Things to fix with functions:
- Return type inference

New concepts:
- def
- lambda syntax
- default is public
- static section
  - funny thing is that
- modules
  - Type inferred for module, interface of exposed members

Bodies of modules, classes, statics are blocks:
-

Modules:
-

Compilation:
- Produces two things:
  - .js file with the implementation
  - Interface file, Strada syntax
    - Enables separate compilation
    -

Two namespaces;
- Type namespace
- Member namespace

Module declaration does two things:
- Defines an interface type named "Math"
- Defines a member named Math in the member namespace

Classes:
- Nest in modules, not in classes
-

Verification and type optimization of classes

Two names Math
- Import Math

Execution:
- Strada.optimize(…)
- Irrespective of what compiled to it

Interfaces:
- Not structurally compatible if they contain classes
  - Seems bad

```
Class Point
{
  Def Foo() {
    Def x = 1;
    Def Bar(this self: IElement, …)
    {
    }
  }
}
```

```
def foo(x:int) => x+1;
def foo = (x:int) => x+1;
def foo : (int) => int = (x:int) { return x + 1; }
```

Type notations:
- [(int,int) => int]
- Int[]
- [int]

Release:
- Strada compiler written in Strada
- Winning in this space is not about protecting IP
- Business is about leadership in this space
- Can meet our objectives without hording IP
- Not to say we can't benefit from secrecy
-

Performance concerns:
- How expensive are instanceof checks?
- Can you afford to make it all public
- Internal?
  - Means internal to a module
-