Notes - 12/02/2013

Monday, December 02, 2013 10:07 AM

- Schedule update

- Bill's email about specializations and merged interfaces]
    - interface I {
        - Foo(f: "foo"): Bar;
        - Foo(f: string): Bar;
    - }
    - interface I {
        - Foo(f: "baz"): Baz;
    - }
    - Interface J {
        - f: Foo;
    - }
    - A couple things:
        - Allowing you to augment members is not allowed by the spec, yet the compiler allows it. Should error
        - Suggestion:
            - Disallow merging of call signatures across interfaces. Need to investigate breakages. If breakages are too much, the workaround is to require users to order their interfaces for merging.
            - lib.d.ts extensions may be tricky
- My email with the remaining spec questions
    - Generic constraints and extending classes (1st most important)
        - Investigating constraints that are recursive
        - This would predicate the kind of checking on whether or not there is a constraint. If it doesn't have a constraint, we don't do any constraint check
        - Get rid of "bullet 1"
        - If we do #5 below, we can remove bullet 4 and half of bullet 3.
            - With #5, we infer the base constraint
        - To investigate:
            - Need to check how often <any> is passed when there are constraints
                - (aka: switching to subtype checking instead of assignment compat)
            - Use of recursive/reflexive constraints in internal/external code
    - Statics/non-inherited. (2nd most important)
        - Anders+Steve this is a high pri
        - Research: revisit if it's a regression
    - Type inference (3rd most important)
        - Concerned that testing and maintaining back compat will be difficult. Anders to spec
    - Covariant arguments are unsound
        - Don't care
    - Recursive constraints (Should be done if we do #3)
        - Bill is investigating
    - Contextually typing return type
        - Not high pri, we could ship without them