Notes - 2/1

Friday, February 1, 2013 10:04 AM

[In 41/2731 at 10:00 today.]

Agenda:

- TC39 meeting updates
  - Modules
  - Statics
  - Classes must extend constructors
- Implementation updates
  - Pull progress
- Handling of structural comparison for generic types
  - Came up with restriction on generative recursion in generic types
  - Example that won't work:
    ```
    interface List<T> {
      a: T;
      n: List<T>;
      owner: List<List<T>>;
    }
    ```
  - Can we make 'Array.prototype.concat' work due to varargs
  -
  - Constraints can reference types they are constraints on
    - Got rid of spec concept of 'upper bound'
    - Stick 'Object' in place of type parameter
  - Contextual typing:
    ```
    Interface Comparable<T> {
        compareTo(other: T): number;
    }
    Interface Comparer {
      <T extends Comparable<T>>(x: T, y: T): number;
    }

    Var f: Comparer = (x,y) => { … } // what type does x have in there
    ```
  - Note: This may be the only inferred non-denotable type currently
  - To do: What do these examples look like in VS

- Modules
  - Exact meaning of 'export ='
    - 'export = expression' you export just a value
    - 'export = class' you get a value and a type

```
m.ts:
export interface Foo { … }
export = class C {
  x: Foo;
}

Other.ts:
Import C = module('m')
New C();
Var x: C;
Var x: C.Foo;
```

<u>Better:</u>

```
function foo(…) { … }
module foo {
  export interface Ifoo { … }
}
export = Foo;
```

m.d.ts:

Point.ts:

```
export = Point;
```

```
class Point {
}
```
// This is approved

- Import syntax (dotted paths?)
    - Internal module aliasing is allowed but doesn't work
- module('foo') syntax
- Multi-file modules

Global:
- a.ts
- b.ts
- c.ts

Island #1:
- x.ts

Island #2:
- y.ts

- Path forward on modules
- On backlog, but probably needs offline working group first:
    - Export =
-
-
-
- Others?

Luke