Notes - 8/2

Friday, August 2, 2013 10:07 AM

[In 18/3555 at 10:00 today.]

Agenda:

- Topics:
  - 'typeof' and .d.ts generation
  - Rules for visibility of interfaces generally
    - Because 'export' is not required, we can't ever put non-exported decalarations inside declare modules

```
Module M {
  module N {
    export var x: typeof N;
  }
 export var y: typeof N;
}

Module A {
  export var x: typeof B;
}
module B {
  export var y: typeof A;
}
export var z: typeof A;

var z: {
  var x: typeof B;
}
```

  - Should instanceof RHS really be requiring callable/constructable

```
p instanceof Point
```

  - Should restriction that 'export =' only refer to something defined in the module be loosened?
    - Yes. Still only a simple identifier.

```
///<reference path="jquery.d.ts" />

declare module "jquery" {
  export = $;
}
```

  - References to modules that contain runtime code but no exports

```
// file1.ts
window.Foo = 34;

//file2.ts
import file1 = require('file1')
var z: require('file1').Point;
console.log(window.Foo)
```

Action: Look at impact this change would have on Monaco and TFS.

  - __extends and accessors on statics
    - Options:
      - No static accessors
      - Ugly code gen
  - Super in statics

```
class Base {
        static foo() {

                }
}


class Foo extends Base {

        static foo() {
                this.foo();
                super.foo(); // allowed according to spec - should be allowed in compiler
        }

        static bar = super.foo(); // should *not* be allowed
        baz = super.foo(); // should be allowed - needs spec update
```

}

- Others?
- Future:
  - ‘export default’ in ES6 modules

Luke