Notes - 7/11 - SENT

Wednesday, July 11, 2012 1:04 PM

#79

Yes – I'll add the optional properties topic to the agenda. Jonathan and I haven't yet completed the investigation of the impact of the larger change to tighten up implicit downcasts, but we can discuss some of the findings so far.

Luke

From: Jonathan Turner
Sent: Wednesday, July 11, 2012 10:30 AM
To: Anders Hejlsberg; Luke Hoban; Strada Design Team
Subject: RE: Strada Design Meeting Agenda - 7/11/2012

Luke, did we also want to talk about the prelim results of disabling implicit downcasts?

Jonathan

From: Anders Hejlsberg
Sent: Wednesday, July 11, 2012 10:28 AM
To: Luke Hoban; Strada Design Team
Subject: RE: Strada Design Meeting Agenda - 7/11/2012

At some point we also need to decide whether to support optional properties and the related changes to assignment compatibility.

```
var x = <Customer[]>[];
Var x = new Customer[];

Var x = <Custoemr>null;
```

Anders

From: Luke Hoban
Sent: Wednesday, July 11, 2012 9:35 AM
To: Strada Design Team
Subject: Strada Design Meeting Agenda - 7/11/2012

[In 41/4749 at 1:00 today.]

Agenda:

- ES6-aligned classes topics
  - Minor deviations from ES6 class proposal [attached]
    - `extends Object should be the same as saying nothing`
    - `You can still explicitly call super("foo") with Object base class`
    - `No class side inheritance for now`
    - `Base class expressions - problem is compiler complexity`
  - Review Zephyr team coding guidelines for ES6 class conversion [attached]
  - Error on missing super call [attached]
    - Yes - super must appear somewhere in the constructor
- Recursive type checking
  - If there's any cycles return any
- Close on enums [attached]
  - Next time
- Optional properties
- Minification
- new Customer[]
- Private constructors - low priority
- Others?

```
Class Foo extends Bar {}

var x = <Foo>bar();

// <Foo> becomes a "type assertion"
// - If it could succeed then it's legal
```

```
function foo() {
 var x = foo();
 return 3;
}
```

```
Function fib(n) {
 if(n==0 || n == 1) return 1;
 var x = fib(n-1) + fib(n-2)
 return x;
}

function bar() {
 return [ bar() ];
}
```

Ideas;

- Go to any instead of error
- Offer compiler mode that errors on inferred any
  - function foo(x) { } //error
- With the above, we could
- ///<style no-any="true"/>

```
class Derived extends Base {}
b
function foo(x: Derived): Base;

foo = function(x: Base){return new Derived(); }  // okay
foo = function(x: Derived){return new Derived(); } // okay
foo = function(x: Base){return new Base(); } // error


var base = Foo(new Derived());
```