Notes - 6/6: SENT

Monday, June 06, 2011 11:11 AM

6/6

Two new keywords – public and private.

- They indicate that things become fields, they can apply to parameters and members

```
// Prototype pattern
class Point(private x: number, private y: number)
{
  public getX() => x
  public getY() => y
  public moveBy(p : Point) => new Point(x + p.x, y + p.y);
}
```

```
// Closure pattern
class Point(x: number, y: number)
{
  public getX = function() => x;
  public getY = function() => y;
  public moveBy = function(p: Point) => new Point(x + p.getX(), y +
p.getY());
}
```

```
// Closure pattern
class Point(public x: number, y: number)
{
  public getX = function() => self.x;
  public getY = function() => y;
  public moveBy = function(p: Point) => new Point(x + p.getX(), y +
p.getY());
}
```

- Fields are explicit
- 'Self'
- What about Point3D

```
class Point3D(x: number, y: number, private z: number): Point(x,y)
{
  public getZ() => z;
  public length() => Math.sqrt(x * x + y * y + z * z);
}
```

- Private means module-private
- Summary:
  - Public and private put things on the instance
  - Access the current instance with 'self'.
  - 'this' has type any, generates warnings, but acts as true this
  - Can do 'funciton(this:Foo)' to shut up warnings and
  - Any parameters and constructor vars are not accessible from "members"

```
var v = {
  f: function() {}
  g: function(this: IMyTypeForV) { this.f(); }
}

var v = new class : IElement {
  var v = asdkfs;
  public f() { }
  public g() { f (); }
}
```