Notes - 2/11: SENT

Saturday, March 12, 2011 5:42 PM

IMPORTANT

Code generation

- Output form
  - Usable from JavaScript
- Only going to get so much leniency in terms of perf on other browsers
- Optimistic
  - Runs as fast as anything on Chrome
    - And obvious that this is true
  - Generally like it
  - Are we overlooking cases where violation is “okay”
  - Violations are:
    - Writing to private fields
    - Passing some other type where a class is expected in a public method
  - Debug setting to throw on violation
  - What does Strada.TypeCheck do?
    - You throw if you aren’t prepared to continue executing
    - Fine to say that error behavior differs
      - Throw on IE
      - Muddle on in Chrome (redacted)
    - Only throw in debug mode?
    - At runtime,
  - Second disadvantage:
    - More work
  - Even further alternative
  - Like the idea that there are typechecks there and if they fail:
    -
    - Have a diagnostic mode
  - Want to *not* do:
    - Incur cost on other browsers
  - Another possibility
    - If you do violate encapsulation – if it’s conformable we could
  - Could Strada provide a set of helpers for
  - Could automagically emit two entry points
    - Once you are in, go to the fast
    - If you aren’t, go to the slow
  - Advantageous to have runtime representation of what you said at compile time
    - Basic reflection
    - Use Object literals
    - 1) You get reflection
    - 2) Runtimes which are hip to this can use the metadata
    - Stick a typedescription property on the object
  - How much reflection do we reify in the model?
    - Lots of metadata kills us in large projects
    - To ultimately get as performant as we need on phones – need to get rid of all this
    - Demand pageable reflection information
  - Fast arrays of primitives
  - Anders:
    - If we can have strongly typed arrays that can be used like List<T>, then forget about generics
    - Need to have a discussion around Arrays
- Minimal metadata mode
- Reflection opt-in
- Metadata:
  - Chrome may allocate it

Class constructors

- Sucinctness

- More like a JavaScript function

```
class Point(int x, int y) {

  string name = "hello";

  public int getX() = x;
  public int getY() = y;

  public void move(Point p) {
     x += p.getX();
     y += p.getY();
  }
  public int getLength() = x * x  + y * y;

}


class Point3D(int x, int y, int z)
   : Point(y,x)
{
   public

}
```