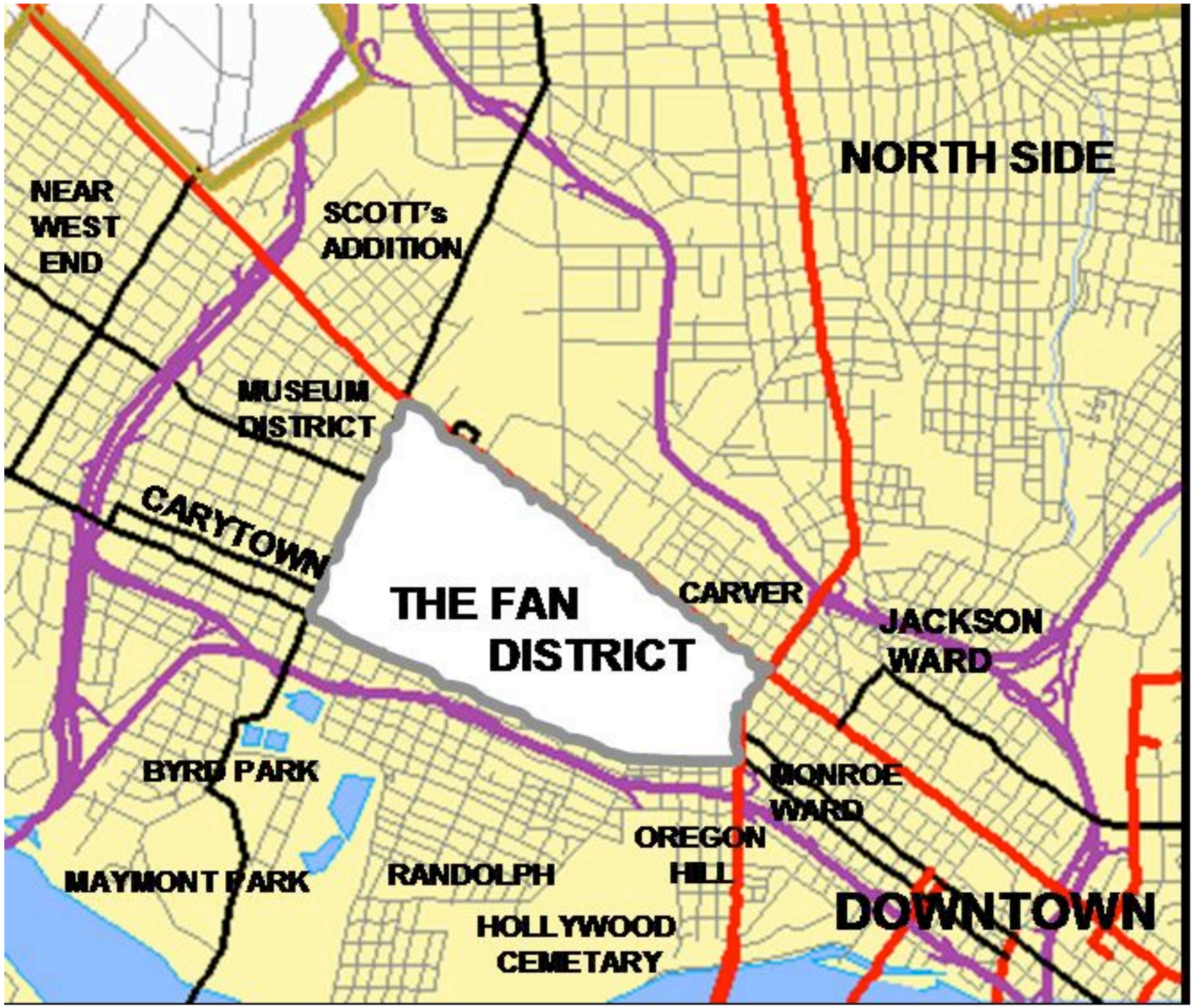
http://fandev.org/
Scott Bale
http://kablooie.puredanger.com/

The Real Fan - Richmond, VA



Hello World

```
class HelloWorld
{
  static Void main()
  {
    echo("hello world")
  }
}
```

Elevator Pitch

- Runs on JVM or CLR
- Hybrid OO/dynamic language
- Both static and dynamic typing

- First class functions (closures)
- Compiles to JavaScript
- Open Source
- No primitivies or checked exceptions

Fibonacci - The Real Hello World

```
class Fibonacci
{
  static Void main()
  {
   echo(fibNum(5))
  }

  static Int fibNum(Int i)
  {
   if (i==0) return 0;
   if (i==1) return 1;
   return fibNum(i-2) + fibNum(i-1)
  }
}
```

Fibonacci with Function

```
class Fibonacci
{
  static Void main()
  {
   echo(fib.call(5))
  }

  const static Func fib := |Int i->Int|{
    if (i==0) return 0;
    if (i==1) return 1;
    // a+b is shortcut for a.plus(b) which is statically typed
    return ((Int)fib.call(i-2)) + ((Int)fib.call(i-1))
  }
}
```

Slightly Cleaner - Dynamic Invocation

```
class Fibonacci
{
  static Void main()
  {
   echo(fib.call(5))
  }

  const static Func fib := |Int i->Int|{
    if (i==0) return 0;
    if (i==1) return 1;
    // below, try a->plus(b) instead to avoid casting
    return fib.call(i-2)->plus(fib.call(i-1))
  }
}
```

Meet the Creators - Brian and Andy Frank

Why Fan? In their own words...

"Fan is designed as a practical programming language to make it easy and fun to get real work done."

"The beauty of a new language is that it gives you a clean slate to fix all the little things that aggravate you (we built Fan to scratch our own itches). "

"Everything we build for SkyFoundry will be written in Fan .... Now we'll be focused on enhancing and using Fan full-time."

Fan Milestones

2005 – Fan begun, designed to be portable between Java and .NET

12/05 – fandev.org launched

12/05 – pure OO type system

2/06 – fcode (deployment portability)

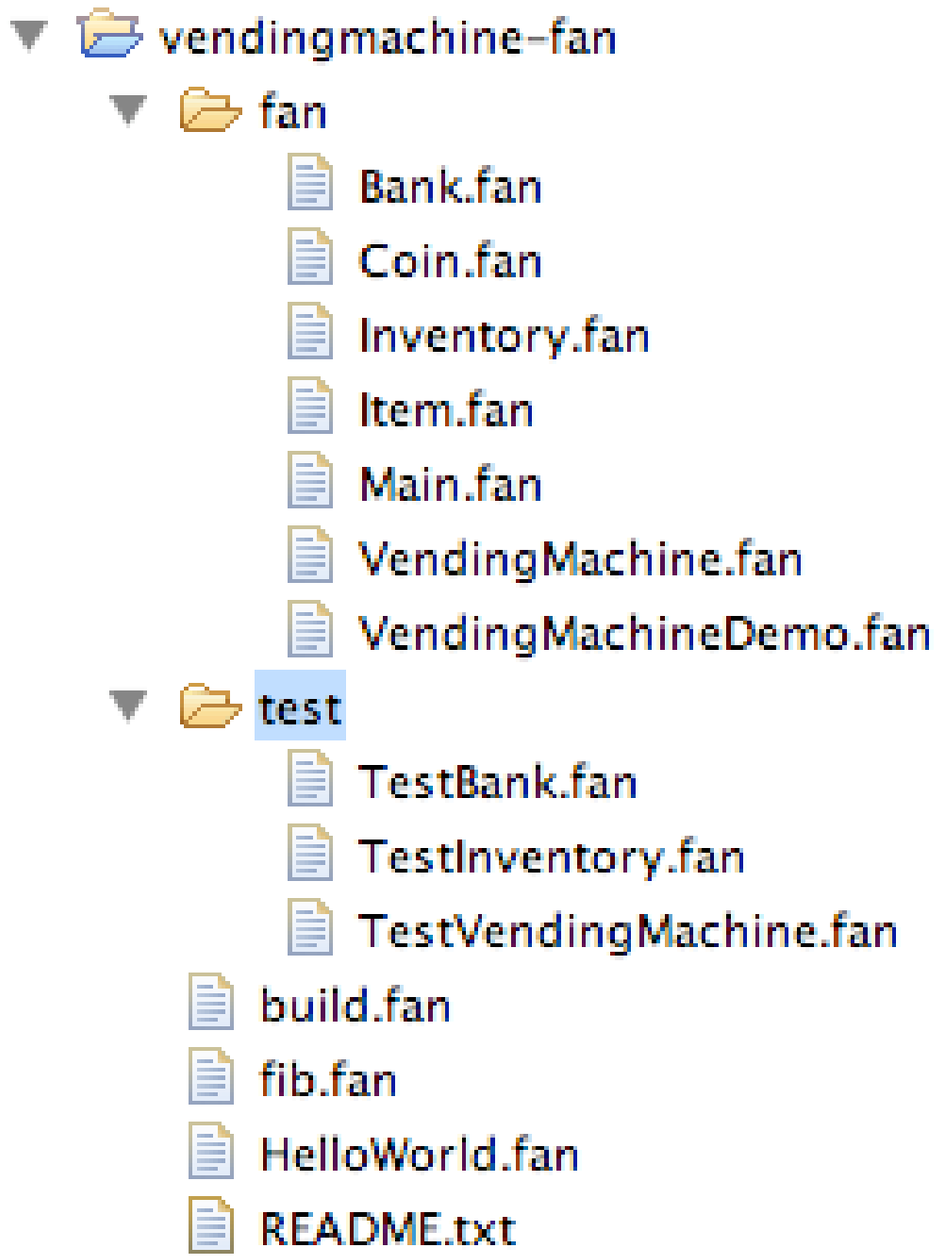
10/06 – compiler re-implemented in Fan

4/08 – fandev.org implemented in Fan

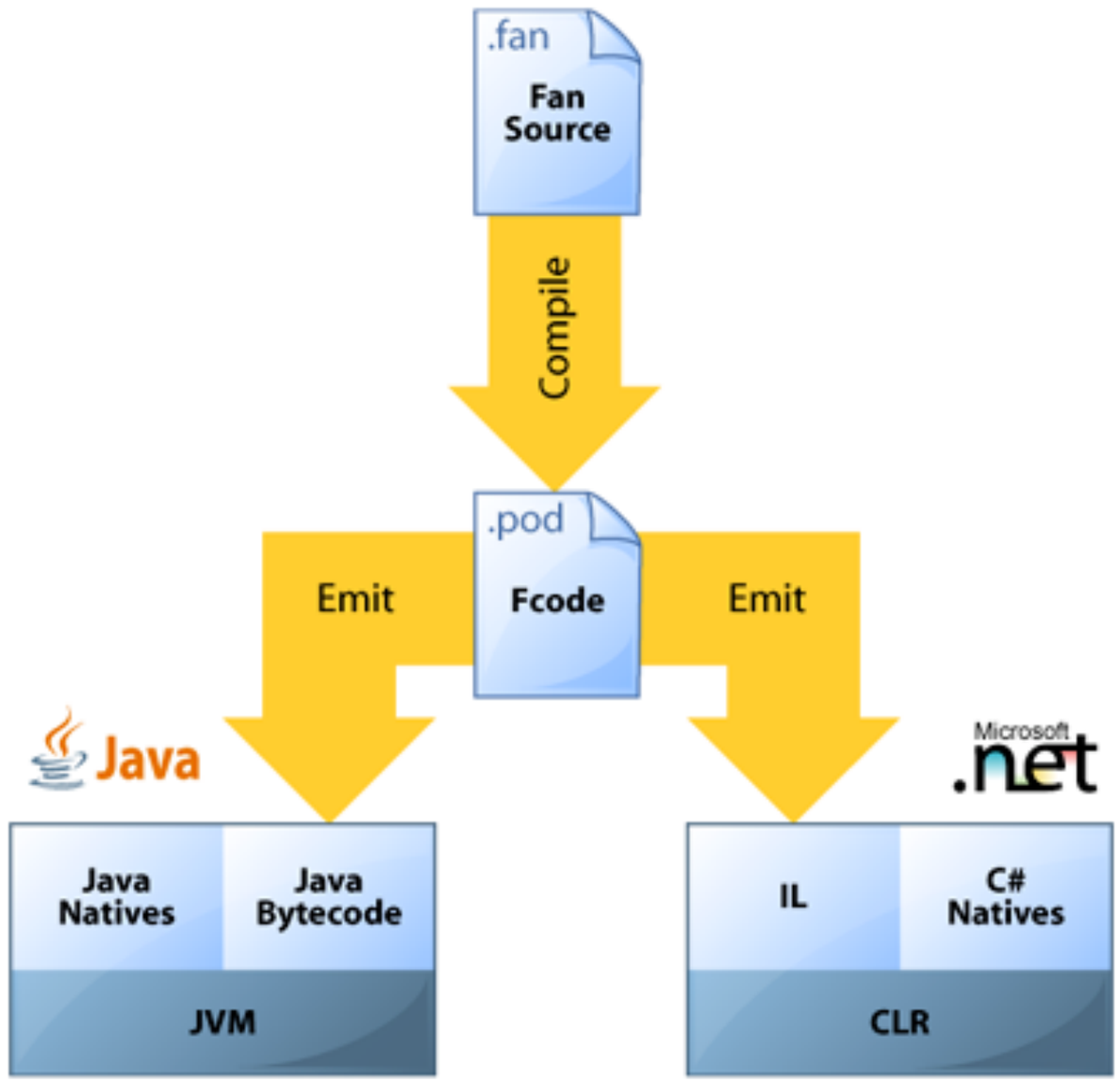
4/08 - .NET env passes test suite

12/08 – SkyFoundry founded

Introducing the Fan Vending Machine



Fan Portability - Binaries and Libraries



A Fan enum - Coin

```
enum Coin
{
  nickel(5),
  dime(10),
  quarter(25),
  dollar(100);

  private new make(Int cents) { this.cents = cents; }

  const Int cents;
}
```

Null-safe types and the Elvis Operator

```
Obj foo := bar.baz.something

Obj? foo := bar?.baz?.something

Obj? foo := bar?->baz?->something
```

Null-safe types Example - Inventory.buy()

```
Item? buy(Item item)
{
    if (inventory.hasItem(item) &&
bank.purchase(item.cost))
    {
        return inventory.remove(item)
    }
    return null
}
```

Function currying with the curry operator &

```
f := |Int a, Int b, Int c->Str| { return "$a $b $c" }
g := &f(0)
f(0, 1, 2)  =>  "0 1 2"
g(1, 2)     =>  "0 1 2"

Void onPress(|,| callback) {...}
Void pressed(Str what) { echo(what) }

ok.onPress(&pressed("ok"))
cancel.onPress(&pressed("cancel"))
```

Example of a closure

```
private Int acceptCoins(Int price)
{
    Int changeDue := coinsToReturn.reduce(0)
        |Obj change, Coin coin->Obj|
        { return (Int)change + coin.cents }
    changeDue -= price
    coins.addAll(coinsToReturn)
    coinsToReturn.clear
    return changeDue
}
```

Dynamic invoke operator ->

The -> operator, behind the scenes, generates a call to sys::Obj.trap()

```
a->x          a.trap("x", [,])
a->x = b      a.trap("x", [b])
a->x(b)       a.trap("x", [b])
a->x(b, c)    a.trap("x", [b, c])
```

Obj.trap() is Fan's "Method-Missing"

```
override Obj? trap(Str name, Obj?[]? args){

        if (name == "info"){
            return "demo: $vm"
        } else if (name == "deposit"){
            vm.deposit(getListFromMap(args[0]))
            return null
        } else if (name == "buy"){
            return vm.buy(args[0])
        } else {
            return super.trap(name, args)
        }
    }
```

Returning correct change

```
private Void makeChange(Int changeDue){
    sortCoins
    coins.eachWhile |Coin coin->Int?|
    {
        if ( coin.cents <= changeDue )
        {
            coinsToReturn.add(coin)
            changeDue -= coin.cents
        }
        return changeDue == 0 ? changeDue : null
    }
    coinsToReturn.each| Coin coin |
{coins.remove(coin) }
```

Links

- http://fandev.org/
- http://twitter.com/briansfrank
- http://twitter.com/afrankvt
- http://github.com/scottbale
- http://kablooie.puredanger.com/