

```

Link = class:
  fields first, rest
  length():
    1 + rest.length()
  end
end

```

"Length" interface

Dynamic Dispatch

```

fun Link(first, rest):
  [dict:
    "length": lam():
      1 + rest.get("length")()
  ]
end

```

```

Empty = class:
  length(): 0 end
end

```

```

fun Empty():
  [dict: "length",
    lam(): 0 end]
end

```

```

lst = Link(4, Link(7, Empty()))
lst.length()

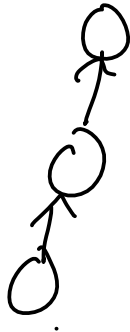
```

o. fld

PROTOTYPE

SELF  
JS

INHERITANCE



lst = { length: lam() ... end }

lst2 = { super: lst,  
fold: lam(f, bar): ... end }

lst3 = { super: lst2,  
map: lam(f): ... end }

lst2.fold(...)

lst2.length()

lst3.length()

fun lookup(o, x):

cases(option) o.get(x):

| some(v) => v

| none =>

cases(option) o.get("super"):

| none => error

| some(s) =>

lookup(s, x)

end

end

end

lookup(lst3, "length")

```

LinkWFold = class extends Link:
  fields first, rest
  fold(f, base):
    f(first, rest.fold(f, base))
  end
end

```

AddFold = fun      (super):

```

fun LinkWFold (first, rest):
  merge(Link(first, rest),
        {dict:
          "fold",
          lam(f, base): ...end})
  end

```

*Diagram: A red arrow points from the boxed `Link(first, rest)` to the `super(first, rest)` call in the dictionary.*

Mixin : Class → Class

```

LinkWFold = AddFold(Link)
AddMap(AddFold(Link))

```

fun      (super):

```

fun EmptyWFold():
  merge(Empty(), super(),
        {dict:
          "fold",
          lam(f, base):
            base
            end})

```

```

EmptyWFold = class extends Empty:
  fold(f, base): base
end

```

CLASS EXTENSION  
INHERITABLE









