DOM Membrane

the trick here is the example from the paper is (almnost) all that's needed:

```
forall S : Set, nd : Node
  [ <will<changes< nd.property>> in S >> -->
  exists  o : Object[
    o ∈ S && !(o : Node) && !(o: Wrapper) &&
      [ exists nd' : Node < o access nd' >  ||
      exists w:Wrapper. exists k:Number.
        (⟨o access w⟩ ∧ nd.parent(k)=w.node.parent(w.height)) ]]
```

that's fine for a one-way wrapper; turns out it would requires a two-way membrane if e.g. the DOM got a notify messaage.

Here's a DOM with membrane

```
type Node  = interface {
   property -> String
   property:=(_ : String)
   parent -> Node
   click
   callback( l : Listener )
}

type Listener = interface {
   clicked(n : Node)
}

def root = object {
  method property { "Root" }
  method property:=(_) { }
  method parent {self}
  method callback( _ ) { }
}

class node(parent' : Node, property' : String) {
  method parent { parent' }
  var property is public := property'
  method callback( l : Listener ) {
    l.clicked( self )
  }
}
```

```
method usingWrappers(unknwn){
  def n1 = node(root,"fixed")
  def n2 = node(n1,"robust")
  def n3 = node(n2,"const")
  def n4 = node(n3,"fluid")
  def n5 = node(n4,"variable")
  def n6 = node(n5,"ethereal")

  def w = n5 //BUG

  def w  = wrapper(n5,1)
  //w.parent.parent.parent.property:= "hacked"

  w.callback( object {
      method clicked(w) { w.parent.parent.parent.property:= "hacked" }
  } )

  assert {n2.property == "robust"}
}

usingWrappers( object { method untrusted( w ) { } } )



class wrapper(node, depth) -> Node {
  //method parent { node.parent }  //BUG
  method parent {
     if (depth > 0) then {wrapper(node.parent, depth - 1)}
        else { error "Hack attempt detected" } }
  method property { node.property }
  method property:=(p) { node.property:= p }
  //method callback( l : Listener ) { node.callback( l ) } // BUG
  //method callback( l : Listener ) { l.clicked( self ) } //SEMI-CHEATING
  method callback( l : Listener ) { node.callback( repparw(l, depth) ) }
}

class repparw( listener, depth ) -> Listener {
   method clicked (node) { listener.clicked( wrapper(node, depth) ) }
}




method assert(block) {
  if (!block.apply) then {Exception.raise "Assertion Failed!"}
}
```

```
method error(string) {
  Exception.raise(string)
}
```