o.x

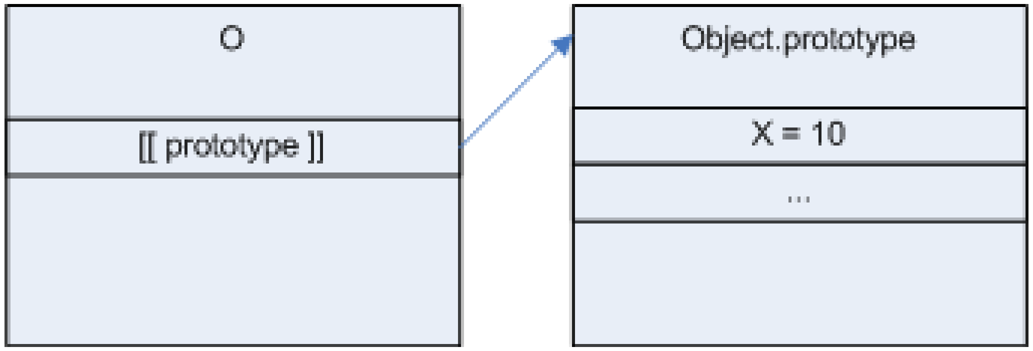
the Managed JScript type system

Pratap Lakshman, JScript

Script code

```
Object.prototype.x = 10;
var o = new Object();
o.x;
```

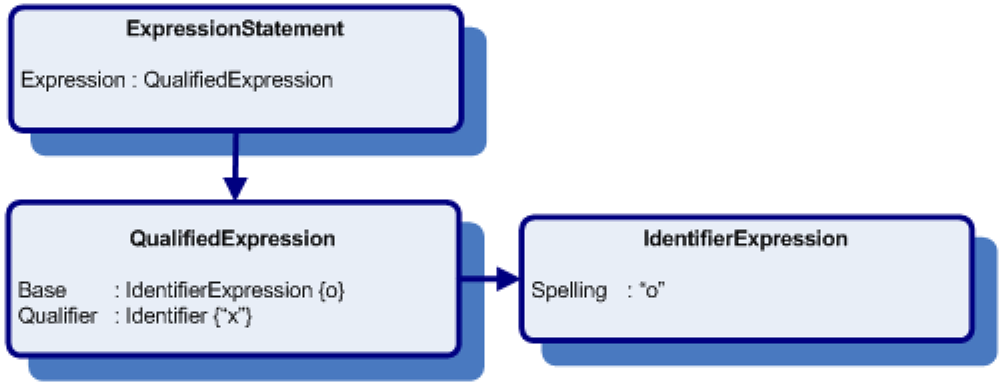
Prototype based inheritance



- The prototype chain of an object is set on construction. Cannot be changed after that.
- The prototype objects are mutable

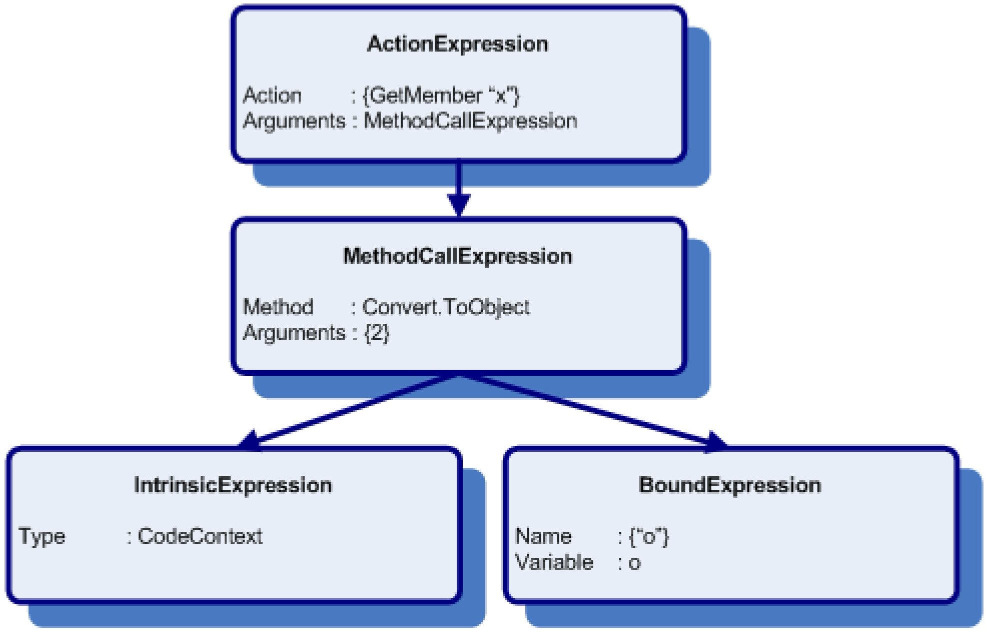
JScript AST

- Tree generated for “o.x”


ExpressionStatement
Expression : QualifiedExpression
QualifiedExpression
Base : IdentifierExpression {o}
Qualifier : Identifier {"x"}
IdentifierExpression
Spelling : "o"

DLR AST

- Tree generated for “o.x”



DLR generated code

```
#SlotStorage3.Invoke(
    Convert.ToObject(
        __global_context,
        o.CurrentValue)
);
```

- #SlotStorage3 represents the action (get) to be performed.
- It captures the details of the action (‘x’ being the member to get).
- Initially the site points back into the DLR runtime to bind a piece of code that can execute this action.

Binding the action

- Get a Rule that captures the details of the action. Including
  - Test: Expression tree that guards the rule against illegal usage
  - Target: Expression tree that represents the action
- Cache the Rule
  - For future fast path usage
- Execute the Rule
  - Generate code for the test & target expressions
  - Make the dynamic site (#SlotStorage3) point to this new piece of code

How does DLR do a get on a JSObject?

- DLR does not know the intricacies of JSObject (prototype chain lookup…).
- JSObject registers with the Action Binder as an entity that can generate Rules for itself.
- DLR delegates the rule generation to our (JS) object.

```
UpdateSiteAndExecute<T>(…, DynamicAction action, object[] args…){
    IDynamicObject ndo = args[0] as IDynamicObject;
      if (ndo != null)
            rule = ndo.GetRule<T>(action, context, args);
}
```

What does the JSObject do?

- Implements the IDynamicObject interface (to generate rules for any action on objects of type JSObject)

```
public interface IDynamicObject {
    StandardRule<T> GetRule<T>(
        DynamicAction action,
        CodeContext context,
        object[] args);
}
```

What is the rule generated for “get”

```
public StandardRule<T> MakeRule() {
  Variable lookupResult = Rule.GetTemporary(typeof(object),"lookupResult");
  Rule.SetTest(
      Rule.MakeTypeTestExpression(Arguments[0].GetType(),  0)
  );
  Rule.SetTarget(
      Ast.IfThenElse(
          Ast.Call(
              Ast.Convert( Rule.Parameters[0],  Arguments[0].GetType() ),
              Action.IsBound ?
                  Arguments[0].GetType().GetMethod("TryGetBoundItem") :
                  Arguments[0].GetType().GetMethod("TryGetItem"),
              Ast.Constant(Action.Name),
              Ast.Read(lookupResult)
          ),
          Rule.MakeReturn(
              Context.LanguageContext.Binder,
              Ast.Read( lookupResult )
          ),
          GetFailureStatement()
      )
  );
  return Rule;
}
```

How does the generated code look?

```
public static object $Microsoft.JScript.Runtime.JSObject.TryGetItem(
  object[] objArray1,
  FastDynamicSite<object, object> site1,
  object obj1
) {
  if ((obj1 != null) && (obj1.GetType() == typeof(JSObject))) {
    object obj2;
    if (((JSObject) obj1).TryGetItem(new SymbolId(0x1000005), out obj2)) {
      return obj2;
    }
    return (UnDefined) objArray1[0];
  }
  return site1.UpdateBindingAndInvoke(obj1);
}
```

Finally, what does TryGetItem do?

```
public virtual bool TryGetItem(SymbolId name, out object value) {
    // Search on this objects dictionary for object specific member
    if (_properties.TryGetValue(name, out value)) {
        return true;
    }

    if (_prototype != null) {
        return _prototype.TryGetItem(name, out value);
    }

    value = UnDefined.Value;
    return false;
}
```

Thank you

Questions

pratapL@microsoft.com

JScript Blog: http://blogs.msdn.com/jscript