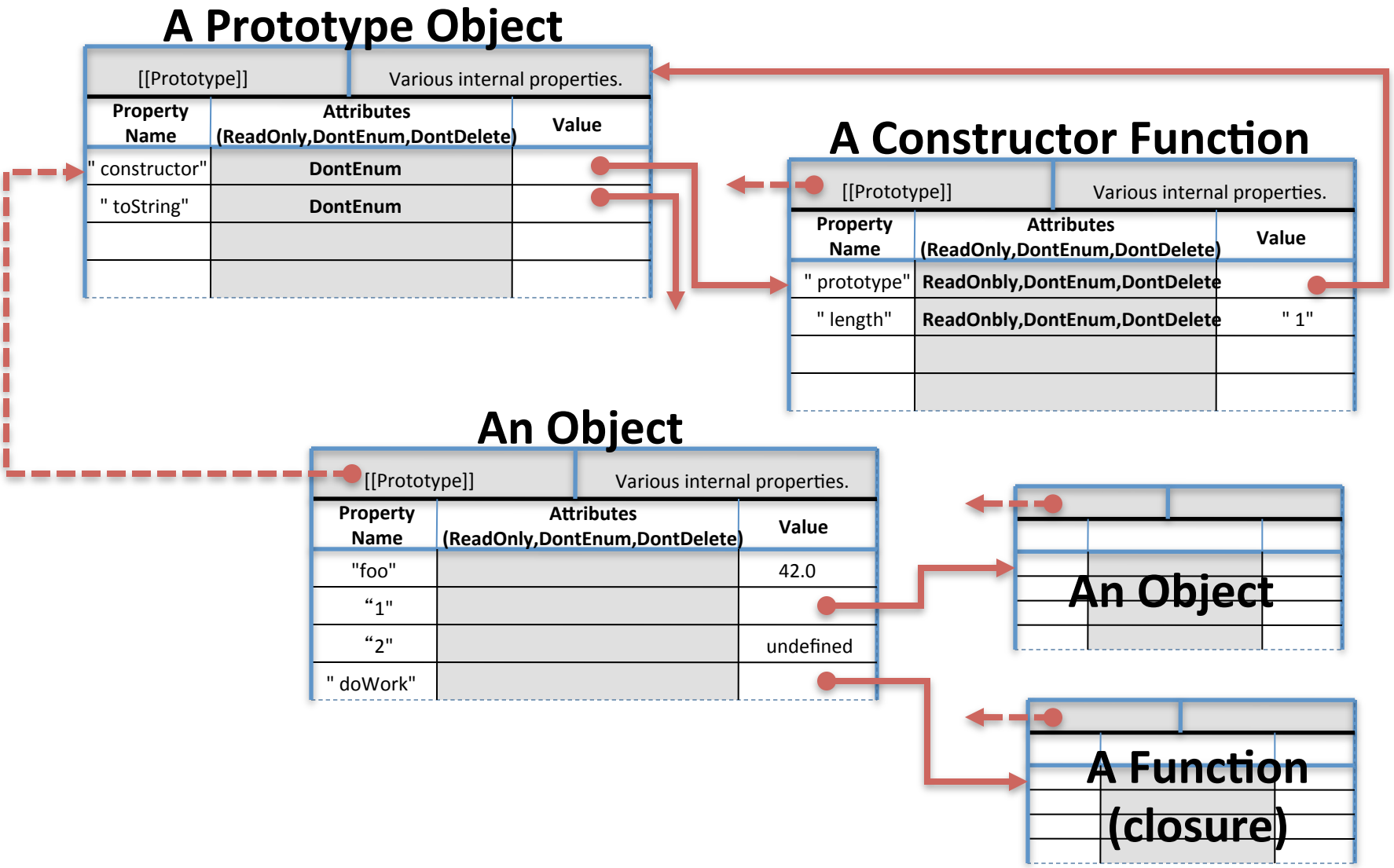
ECMAScript 3.1 Object Model

Allen Wirfs-Brock
Microsoft

ECMAScript 3 Object Model



ES3.1 Object Model Changes

- Rename/repurpose attributes
  - ReadOnly → Writable
  - DontEnum -> Enumerable
  - DontDelete -> Configurable
- Attribute values are programmatically settable and testable
- An object may be programmatically marked as "nonExtensible" (properties may not be added)
- Accessor Properties (getter/setter)

Configurable Attribute

- The [[configurable]] attribute of a property controls whether the definition of an attribute can be programmatically changed:
  - Delete the attribute
  - Change the state of a property attribute: writable, enumerable, configurable
  - Change/delete the getter and/or setter function of an accessor property.
  - Convert a data property to an accessor property or visa versa.
- If Configurable attribute is false for a property
  - None of the above can occur
  - Writable can be change from true to false

Manipulating Properties and Attributes

- "Static" functions accessed via Object constructor
  - Object.defineProperty(obj, propName, propDescriptor)
  - Define a property:

```
Object.defineProperty(o, "length", {
        getter: function() {return this.computeLength()},
    setter: function(value){this.changeLength(value)}  });
    Object.defineProperty(o, "1", {value: 1,
                                   enumerable: true, configurable: true});
```

  - Modify property attributes

```
Object.defineProperty(Array.prototype, "forEach",
                      {enumerable: false, writable:false, configurable: false});
```

Retrieving a Property Definition

- Object.getOwnPropertyDescriptor (obj, propName)

  var desc = Object. getOwnPropertyDescriptor(o, “length));

- Return value isdescriptor with data properties
  - value, writeable, enumerable, configurable
    or
  - getter, setter, enumerable, configurable

- Return value is usable as 3rd argument to Object.defineProperty

Object Lock-down

- Prevent adding properties to an object
  - Object.preventExtensions(obj)
- Prevent adding or reconfiguring properties
  - Object.seal(obj)
- Prevent adding, reconfiguring, modify the value of properties
  - Object.freeze(obj)

Other Object Meta Methods

- Object.defineProperties(obj, propName, descriptorSet)
- Object.create(protoObj, descriptorSet);
- Object.getOwnPropertyNames(obj)
- Object.getPrototypeOf(obj)
- Object.isExtensible(obj)
- Object.isSealed(obj)
- Object.isFrozen(obj)

Example

```
// a Point “class”.
function Point(x, y) {
   const self = Object.create(Point.prototype, {
                toString: {value: Object.freeze(function() {
                                    return '<' + self.getX() + ',' + self.getY() + '>')}},
                          enumerable: true},
                getX: {value: Object.freeze(function() {return x}),
                          enumerable: true},
                 getY: {value: Object.freeze(function() {return y}),
                          enumerable: true}
                 });
   return self;
}
```

Example

```
//Point as a non-final non-abstract root/mixin class where toString is a final method:
function PointMixin(self, x, y) {
   Object.defineProperties(self, {
                toString: {value: Object.freeze(function() { return'<' + self.getX() + ',' + self.getY() + '>')}},
                      enumerable: true},
                getX: {value: Object.freeze(function() {return x}),
                       enumerable: true, flexible: true},
                getY: {value: Object.freeze(function() {return y}),
                       enumerable: true, flexible: true}
                });
}
function Point(x, y) {
   const self = Object.create(Point.prototype); // only for instanceof
   PointMixin(self, x, y);
   return Object.freeze(self);
}
Object.freeze(PointMixin);
Object.freeze(Point.prototype);
Object.freeze(Point);
```