Update on Object Literal Extensions for ES.next

Allen Wirfs-Brock

Mozilla

September 26, 2011

Removed

- Attribute modifiers:

```
{!x: 5,   //non-configurage
  ~y: 6    //non-enumerable
}
```

  - Why? Too much negative community reaction
  - Still includes non-writable properties:

```
{z := 7}
```

- Object modifiers:

```
{.seal,    //or .freeze
  x: 5,
  y: 6
}
```

  - Why? No love. Odd syntax. Added complexity:

```
Object.seal({x: 5,
                    y: 6
});
```

Added Evaluated Property Names

- In an object literal any property name position may be a bracketed expression.
- Actual property name is determine at construction time by evaluating expression.
- It may evaluate to a private name object.

```
const foo = Name.create();
{
  [foo] : 1,
  get [1+2]() {},
  ['prefix'+i++]() {}
}
```

Added

- Comma is optional following method and accessor properties:
- {
  get foo() {return this.computeFoo()}
  computeFoo() {return this[privateFoo]}
  [privateFoo]: 0
  }

Issues

- Attributes for constant and method properties?
  - enumerable: false, writable: false, configurable: false
  - To much lock down?
- Single identifier properties are future-hostile for block lambdas:

```
{foo}   //is it an object literal or a block lambda
```

- Generators and concise methods?

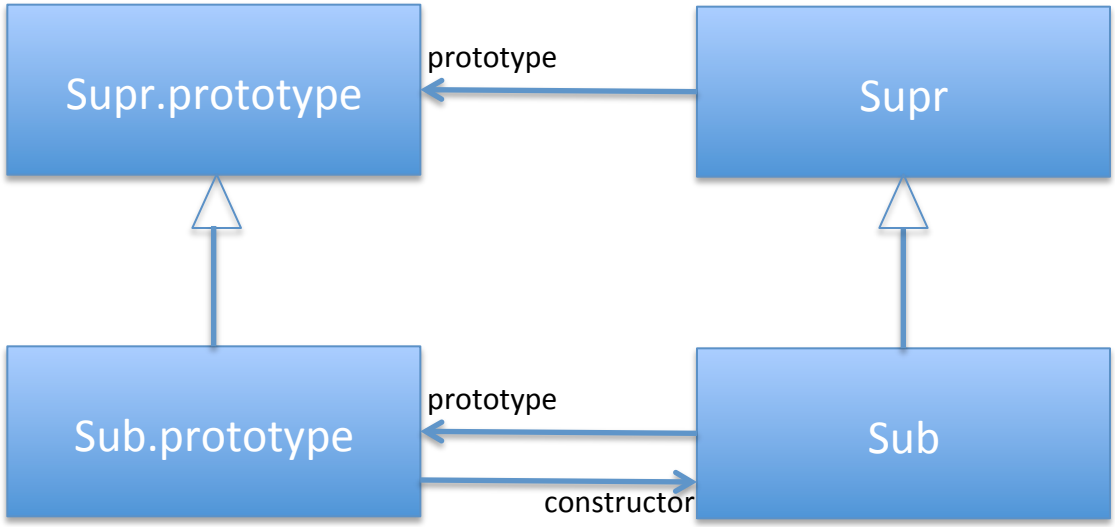
```
{*foo() {… yield bar;…}
}
```

Modified

- Semantics of <| when RHS is function

  let Sub = Supr <| function() {};

- If ('prototype' in Supr):



Added

- Object Extension Literals

  obj.{a:1,b:2,c:3};

- Means approximately:

```
Object.defineProperties(obj, {
  a: {value: 1, enumerable: true, writable: true, configurable: true},
  b: {value: 2, enumerable: true, writable: trueconfigurable: true},
  c: {value: 3, enumerable: true, writable: true, configurable: true}
};
```

- Not a copy operation!
  - Important distinction for **super**
  - Private named properties
- Other syntax possibilities:

  .={ .+{

A “Class” Definition Pattern

```
const className = superClass <| function(/*constructor parameters */) {
    //constructor body
   super.constructor(/*arguments to super constructor */);
   this.{
      //per instance property definitions
   };
}.prototype.{
    //instance properties defined on prototype
}.constructor.{
   //class (ie, constructor) properties
};
```

https://github.com/allenwb/ESnext-experiments

Initializers in Binding Destructurings

```
let {
    a: b = 5,
    b: {x:x,y:y=6} = foo,
    c:  [q,r,s,…t]
     }  = getObj();
```