

The JavaScript Language (Part 3)

Copyright © 2024 by
Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
 - A subset of JavaScript...
 - That is appropriate for COS 333...
 - Through example programs

Agenda

- **Objects (review)**
- Prototypes
- Delegation to prototypes
- Classes

Objects (Review)

- Recall **fraction2.js**, **fraction2client.js**...

```
$ node fraction2client.js
Numerator 1: 1
Denominator 1: 2
Numerator 2: 3
Denominator 2: 4
f1: 1/2
f2: 3/4
f1 is not identical to f2
f1 is less than f2
-f1: -1/2
f1 + f2: 5/4
f1 - f2: -1/4
f1 * f2: 3/8
f1 / f2: 2/3
$
```

fraction2.js (Page 1 of 2)

```

1: //-----
2: // fraction2.js
3: // Author: Bob Dondero
4: //-----
5:
6: 'use strict';
7:
8: const euclid = require('./euclid.js');
9:
10: function createFraction(num=0, den=1)
11: {
12:   if (arguments.length > 2)
13:     throw new Error('Too many arguments');
14:
15:   if (den === 0)
16:     throw new Error('Denominator cannot be zero');
17:
18:   let f = {};
19:
20:   f._num = num;
21:   f._den = den;
22:
23:   if (f._den < 0) {
24:     f._num *= -1;
25:     f._den *= -1;
26:   }
27:   if (f._num === 0)
28:     f._den = 1;
29:   else {
30:     let gcden = euclid.gcd(f._num, f._den);
31:     f._num /= gcden;
32:     f._den /= gcden;
33:   }
34:
35:   f.toString = function() {
36:     return String(this._num) + '/' + String(this._den);
37:   };
38:
39:   f.compareTo = function(other) {
40:     if ((this._num * other._den) < (other._num * this._den))
41:       return -1;
42:     if ((this._num * other._den) > (other._num * this._den))
43:       return 1;
44:     return 0;
45:   };
46:
47:   f.negate = function() {
48:     return createFraction(-this._num, this._den);
49:   };
50:
51:   f.add = function(other) {
52:     let newNum = (this._num * other._den) + (other._num * this._den);
53:     let newDen = this._den * other._den;
54:     return createFraction(newNum, newDen);
55:   };
56:
57:   f.subtract = function(other) {
58:     let newNum = (this._num * other._den) - (other._num * this._den);
59:     let newDen = this._den * other._den;
60:     return createFraction(newNum, newDen);
61:   };
62:
63:   f.multiply = function(other) {
64:     let newNum = this._num * other._num;
65:     let newDen = this._den * other._den;

```

fraction2.js (Page 2 of 2)

```

66:     return createFraction(newNum, newDen);
67:   };
68:
69:   f.divide = function(other) {
70:     let newNum = this._num * other._den;
71:     let newDen = this._den * other._num;
72:     return createFraction(newNum, newDen);
73:   };
74:
75:   return f;
76: }
77:
78: module.exports = { createFraction };

```

fraction2client.js (Page 1 of 2)

```

1: //-----
2: // fraction2client.js
3: // Author: Bob Dondero
4: //-----
5:
6: 'use strict';
7:
8: const readlineSync = require('readline-sync');
9: const fraction = require('./fraction2.js');
10:
11: //-----
12:
13: function readInt(prompt) {
14:   let line = readlineSync.question(prompt);
15:   if (line === '')
16:     throw new Error('Missing integer');
17:   if (isNaN(line))
18:     throw new Error('Not a number');
19:   let n = Number(line);
20:   if (!Number.isInteger(n))
21:     throw new Error('Not an integer');
22:   return n;
23: }
24:
25: //-----
26:
27: function main() {
28:   try {
29:     let n1 = readInt('Numerator 1: ');
30:     let d1 = readInt('Denominator 1: ');
31:     let n2 = readInt('Numerator 2: ');
32:     let d2 = readInt('Denominator 2: ');
33:
34:     let f1 = fraction.createFraction(n1, d1);
35:     let f2 = fraction.createFraction(n2, d2);
36:
37:     process.stdout.write('f1: ' + f1.toString() + '\n');
38:     process.stdout.write('f2: ' + String(f2) + '\n');
39:
40:     if (f1 === f2)
41:       process.stdout.write('f1 is identical to f2\n');
42:     else
43:       process.stdout.write('f1 is not identical to f2\n');
44:
45:     let compare = f1.compareTo(f2);
46:     if (compare < 0)
47:       process.stdout.write('f1 is less than f2\n');
48:     if (compare > 0)
49:       process.stdout.write('f1 is greater than f2\n');
50:     if (compare === 0)
51:       process.stdout.write('f1 is equal to f2\n');
52:
53:     let f3;
54:
55:     f3 = f1.negate();
56:     process.stdout.write('-f1: ' + String(f3) + '\n');
57:
58:     f3 = f1.add(f2);
59:     process.stdout.write('f1 + f2: ' + String(f3) + '\n');
60:
61:     f3 = f1.subtract(f2);
62:     process.stdout.write('f1 - f2: ' + String(f3) + '\n');
63:
64:     f3 = f1.multiply(f2);
65:     process.stdout.write('f1 * f2: ' + String(f3) + '\n');

```

fraction2client.js (Page 2 of 2)

```

66:
67:     f3 = f1.divide(f2);
68:     process.stdout.write('f1 / f2: ' + String(f3) + '\n');
69:   }
70:   catch (e) {
71:     process.stderr.write(String(e) + '\n');
72:   }
73: }
74:
75: if (require.main === module)
76:   main();

```

Objects (Review)

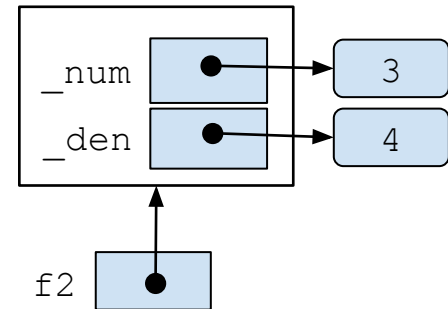
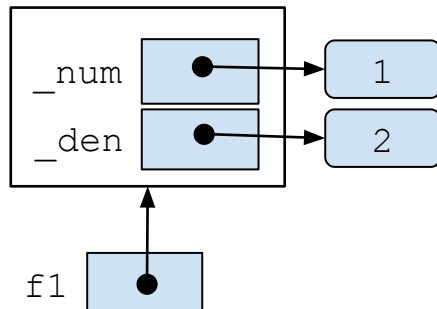
In Python

```
f1 = Fraction(1, 2)  
f2 = Fraction(3, 4)
```

```
add(self, other):  
    ...
```

```
sub(self, other):  
    ...
```

...



Explicit `self` parameter allows `Fraction` objects to share same function defs

Objects (Review)

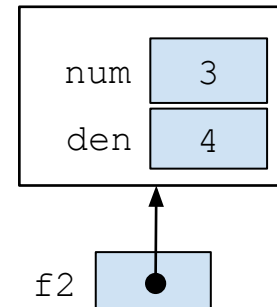
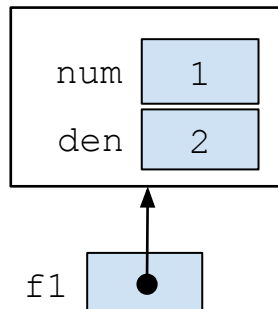
```
Fraction f1 = new Fraction(1, 2);  
Fraction f2 = new Fraction(3, 4);
```

In Java

```
add(this, other)  
{...}
```

```
sub(this, other)  
{...}
```

...

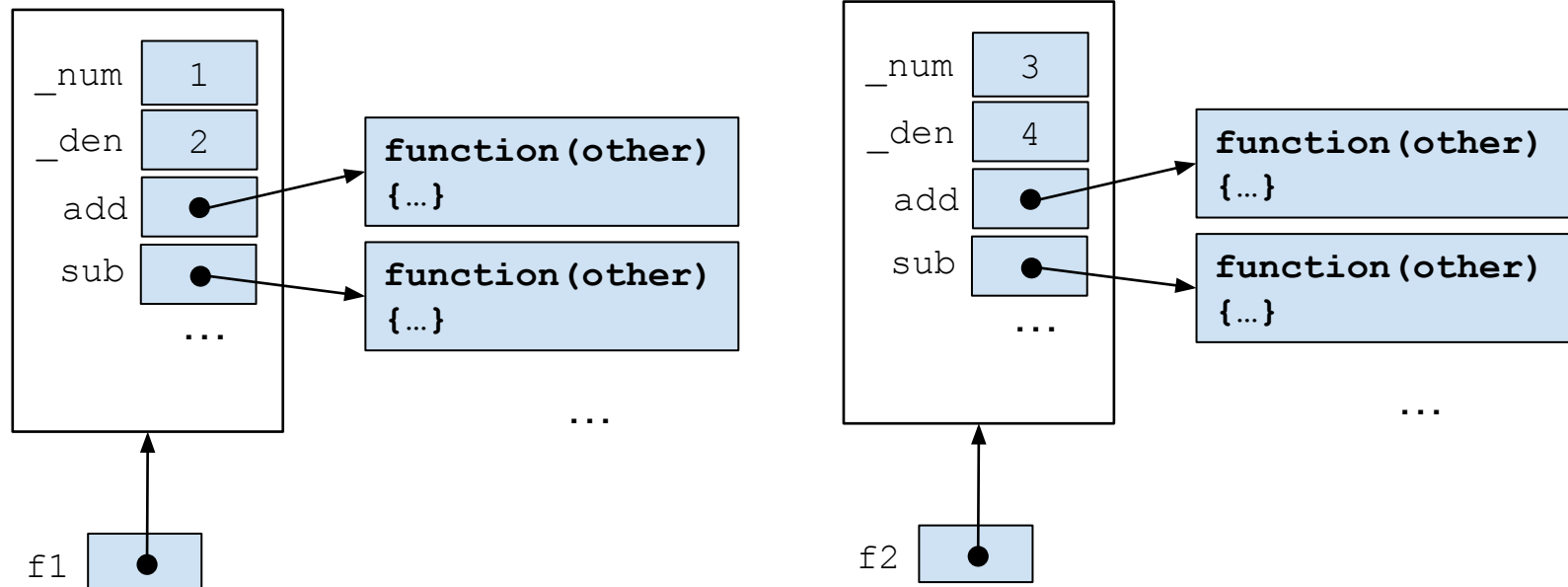


Implicit `this` parameter allows `Fraction` objects to share same method defs

Objects (Review)

In JavaScript (so far)

```
let f1 = createFraction(1, 2);  
let f2 = createFraction(3, 4);
```



Objects (Review)

- **Solution (part 1)**

- ...

Agenda

- Objects (review)
- **Prototypes**
- Delegation to prototypes
- Classes

Prototypes

- See **fraction3.js**, **fraction3client.js**

```
$ node fraction3client.js
Numerator 1: 1
Denominator 1: 2
Numerator 2: 3
Denominator 2: 4
f1: 1/2
f2: 3/4
f1 is not identical to f2
f1 is less than f2
-f1: -1/2
f1 + f2: 5/4
f1 - f2: -1/4
f1 * f2: 3/8
f1 / f2: 2/3
$
```

fraction3.js (Page 1 of 2)

```

1: //-----
2: // fraction3.js
3: // Author: Bob Dondero
4: //-----
5:
6: 'use strict';
7:
8: const euclid = require('./euclid.js');
9:
10: function Fraction(num=0, den=1) {
11:   if (arguments.length > 2)
12:     throw new Error('Too many arguments');
13:
14:   if (den === 0)
15:     throw new Error('Denominator cannot be zero');
16:
17:   this._num = num;
18:   this._den = den;
19:
20:   if (this._den < 0) {
21:     this._num *= -1;
22:     this._den *= -1;
23:   }
24:   if (this._num === 0)
25:     this._den = 1;
26:   else {
27:     let gcden = euclid.gcd(this._num, this._den);
28:     this._num /= gcden;
29:     this._den /= gcden;
30:   }
31:
32:   this.toString = function() {
33:     return String(this._num) + '/' + String(this._den);
34:   };
35:
36:   this.compareTo = function(other) {
37:     if ((this._num * other._den) < (other._num * this._den))
38:       return -1;
39:     if ((this._num * other._den) > (other._num * this._den))
40:       return 1;
41:     return 0;
42:   };
43:
44:   this.negate = function() {
45:     return new Fraction(-this._num, this._den);
46:   };
47:
48:   this.add = function(other) {
49:     let newNum = (this._num * other._den) + (other._num * this._den);
50:     let newDen = this._den * other._den;
51:     return new Fraction(newNum, newDen);
52:   };
53:
54:   this.subtract = function(other) {
55:     let newNum = (this._num * other._den) - (other._num * this._den);
56:     let newDen = this._den * other._den;
57:     return new Fraction(newNum, newDen);
58:   };
59:
60:   this.multiply = function(other) {
61:     let newNum = this._num * other._num;
62:     let newDen = this._den * other._den;
63:     return new Fraction(newNum, newDen);
64:   };
65:

```

fraction3.js (Page 2 of 2)

```

66:   this.divide = function(other) {
67:     let newNum = this._num * other._den;
68:     let newDen = this._den * other._num;
69:     return new Fraction(newNum, newDen);
70:   };
71: }
72:
73: module.exports = { Fraction };

```

fraction3client.js (Page 1 of 2)

```

1: //-----
2: // fraction3client.js
3: // Author: Bob Dondero
4: //-----
5:
6: 'use strict';
7:
8: const readlineSync = require('readline-sync');
9: const fraction = require('./fraction3.js');
10:
11: //-----
12:
13: function readInt(prompt) {
14:   let line = readlineSync.question(prompt);
15:   if (line === '')
16:     throw new Error('Missing integer');
17:
18:   let n = Number(line);
19:   if (! Number.isInteger(n))
20:     throw new Error('Not an integer');
21:
22:   return n;
23: }
24:
25: //-----
26:
27: function main() {
28:   try {
29:     let n1 = readInt('Numerator 1: ');
30:     let d1 = readInt('Denominator 1: ');
31:     let n2 = readInt('Numerator 2: ');
32:     let d2 = readInt('Denominator 2: ');
33:
34:     let f1 = new fraction.Fraction(n1, d1);
35:     let f2 = new fraction.Fraction(n2, d2);
36:
37:     process.stdout.write('f1: ' + f1.toString() + '\n');
38:     process.stdout.write('f2: ' + String(f2) + '\n');
39:
40:     if (f1 === f2)
41:       process.stdout.write('f1 is identical to f2\n');
42:     else
43:       process.stdout.write('f1 is not identical to f2\n');
44:
45:     let compare = f1.compare(f2);
46:     if (compare < 0)
47:       process.stdout.write('f1 is less than f2\n');
48:     if (compare > 0)
49:       process.stdout.write('f1 is greater than f2\n');
50:     if (compare === 0)
51:       process.stdout.write('f1 is equal to f2\n');
52:
53:     let f3;
54:
55:     f3 = f1.negate();
56:     process.stdout.write('-f1: ' + String(f3) + '\n');
57:
58:     f3 = f1.add(f2);
59:     process.stdout.write('f1 + f2: ' + String(f3) + '\n');
60:
61:     f3 = f1.subtract(f2);
62:     process.stdout.write('f1 - f2: ' + String(f3) + '\n');
63:
64:     f3 = f1.multiply(f2);
65:     process.stdout.write('f1 * f2: ' + String(f3) + '\n');

```

fraction3client.js (Page 2 of 2)

```

66:
67:     f3 = f1.divide(f2);
68:     process.stdout.write('f1 / f2: ' + String(f3) + '\n');
69:   }
70:   catch (e) {
71:     process.stderr.write(e + '\n');
72:   }
73: }
74:
75: if (require.main === module)
76:   main();

```

Prototypes

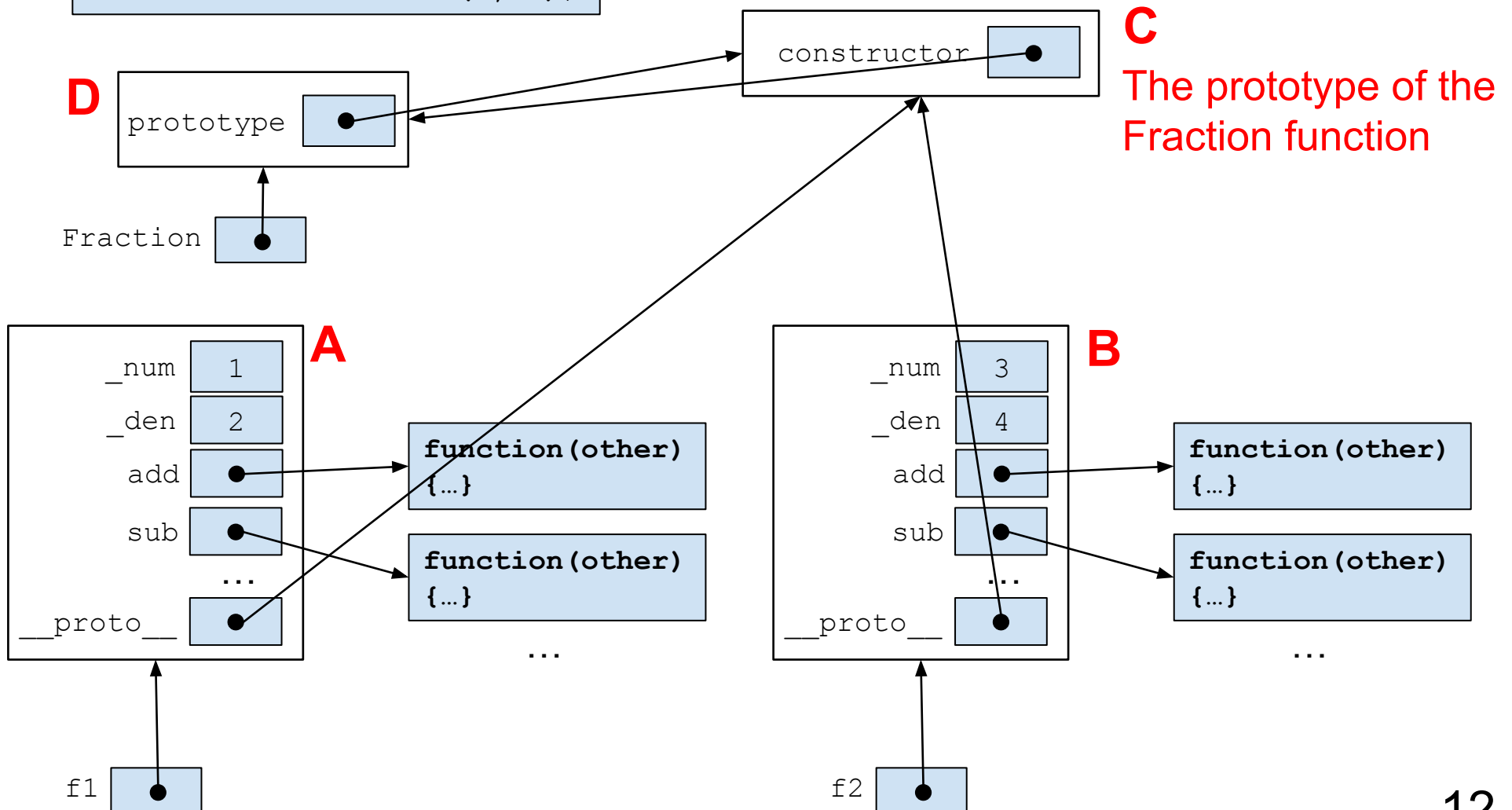
. Prototypes

- Any function has a prototype
 - E.g.: `Fraction` has a prototype referenced by `Fraction.prototype`
- When an object is created by calling a constructor function with a `new` operator, the object has a property named `__proto__`
 - E.g.: `f1` has a `__proto__` property
- The `__proto__` property references the prototype of the constructor function
 - E.g.: `f1.__proto__` references the `Fraction` prototype

Prototypes

In JavaScript (so far)

```
let f1 = new Fraction(1, 2);  
let f2 = new Fraction(3, 4);
```



Prototypes

- **Solution (part 1):**
 - Prototypes
- **Solution (part 2):**
 - ...

Agenda

- Objects (review)
- Prototypes
- **Delegation to prototypes**
- Classes

Delegation to Prototypes

- See **fraction4.js**, **fraction4client.js**

```
$ node fraction4client.js
Numerator 1: 1
Denominator 1: 2
Numerator 2: 3
Denominator 2: 4
f1: 1/2
f2: 3/4
f1 is not identical to f2
f1 is less than f2
-f1: -1/2
f1 + f2: 5/4
f1 - f2: -1/4
f1 * f2: 3/8
f1 / f2: 2/3
$
```

fraction4.js (Page 1 of 2)

```

1: //-----
2: // fraction4.js
3: // Author: Bob Dondero
4: //-----
5:
6: 'use strict';
7:
8: const euclid = require('./euclid.js');
9:
10: function Fraction(num=0, den=1) {
11:   if (arguments.length > 2)
12:     throw new Error('Too many arguments');
13:
14:   if (den === 0)
15:     throw new Error('Denominator cannot be zero');
16:
17:   this._num = num;
18:   this._den = den;
19:
20:   if (this._den < 0) {
21:     this._num *= -1;
22:     this._den *= -1;
23:   }
24:   if (this._num === 0)
25:     this._den = 1;
26:   else {
27:     let gcden = euclid.gcd(this._num, this._den);
28:     this._num /= gcden;
29:     this._den /= gcden;
30:   }
31: }
32:
33: Fraction.prototype.toString = function() {
34:   return String(this._num) + '/' + String(this._den);
35: };
36:
37: Fraction.prototype.compareTo = function(other) {
38:   if ((this._num * other._den) < (other._num * this._den)) return -1;
39:   if ((this._num * other._den) > (other._num * this._den)) return 1;
40:   return 0;
41: };
42:
43: Fraction.prototype.negate = function() {
44:   return new Fraction(-this._num, this._den);
45: };
46:
47: Fraction.prototype.add = function(other) {
48:   let newNum = (this._num * other._den) + (other._num * this._den);
49:   let newDen = this._den * other._den;
50:   return new Fraction(newNum, newDen);
51: };
52:
53: Fraction.prototype.subtract = function(other) {
54:   let newNum = (this._num * other._den) - (other._num * this._den);
55:   let newDen = this._den * other._den;
56:   return new Fraction(newNum, newDen);
57: };
58:
59: Fraction.prototype.multiply = function(other) {
60:   let newNum = this._num * other._num;
61:   let newDen = this._den * other._den;
62:   return new Fraction(newNum, newDen);
63: };
64:
65: Fraction.prototype.divide = function(other) {

```

fraction4.js (Page 2 of 2)

```

66:   let newNum = this._num * other._den;
67:   let newDen = this._den * other._num;
68:   return new Fraction(newNum, newDen);
69: };
70:
71: module.exports = { Fraction };

```

fraction4client.js (Page 1 of 2)

```

1: //-----
2: // fraction4client.js
3: // Author: Bob Dondero
4: //-----
5:
6: 'use strict';
7:
8: const readline = require('readline-sync');
9: const fraction = require('./fraction4.js');
10:
11: //-----
12:
13: function readInt(prompt) {
14:   let line = readline.question(prompt);
15:   if (line === '')
16:     throw new Error('Missing integer');
17:
18:   let n = Number(line);
19:   if (! Number.isInteger(n))
20:     throw new Error('Not an integer');
21:
22:   return n;
23: }
24:
25: //-----
26:
27: function main() {
28:   try {
29:     let n1 = readInt('Numerator 1: ');
30:     let d1 = readInt('Denominator 1: ');
31:     let n2 = readInt('Numerator 2: ');
32:     let d2 = readInt('Denominator 2: ');
33:
34:     let f1 = new fraction.Fraction(n1, d1);
35:     let f2 = new fraction.Fraction(n2, d2);
36:
37:     process.stdout.write('f1: ' + f1.toString() + '\n');
38:     process.stdout.write('f2: ' + String(f2) + '\n');
39:     process.stdout.write('f2: ' + f2 + '\n');
40:
41:     if (f1 === f2)
42:       process.stdout.write('f1 is identical to f2\n');
43:     else
44:       process.stdout.write('f1 is not identical to f2\n');
45:
46:     let compare = f1.compareTo(f2);
47:     if (compare < 0)
48:       process.stdout.write('f1 is less than f2\n');
49:     if (compare > 0)
50:       process.stdout.write('f1 is greater than f2\n');
51:     if (compare === 0)
52:       process.stdout.write('f1 is equal to f2\n');
53:
54:     let f3;
55:
56:     f3 = f1.negate();
57:     process.stdout.write('-f1: ' + String(f3) + '\n');
58:
59:     f3 = f1.add(f2);
60:     process.stdout.write('f1 + f2: ' + String(f3) + '\n');
61:
62:     f3 = f1.subtract(f2);
63:     process.stdout.write('f1 - f2: ' + String(f3) + '\n');
64:
65:     f3 = f1.multiply(f2);

```

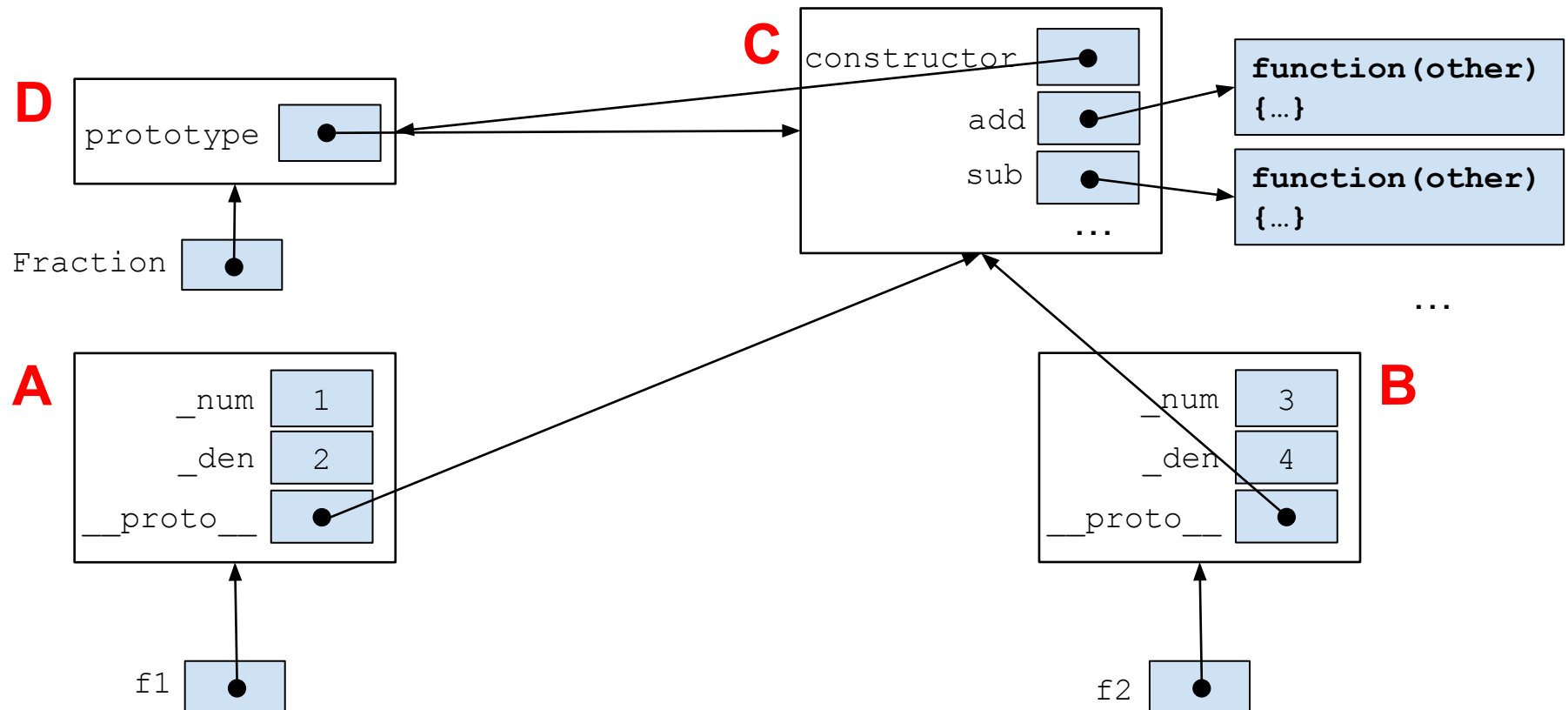
fraction4client.js (Page 2 of 2)

```

66:     process.stdout.write('f1 * f2: ' + String(f3) + '\n');
67:
68:     f3 = f1.divide(f2);
69:     process.stdout.write('f1 / f2: ' + String(f3) + '\n');
70:   }
71:   catch (e) {
72:     process.stderr.write(e + '\n');
73:   }
74: }
75:
76: if (require.main === module)
77:   main();

```

Delegation to Prototypes



`f1.add(f2)` \Rightarrow runtime looks for `f1.add()` then `f1.__proto__.add()`
`f2.add(f1)` \Rightarrow runtime looks for `f2.add()` then `f2.__proto__.add()`

Classes

- **Problem**

- Delegation to prototypes is distant from mainstream OOP
- Difficult to learn & understand

- **Solution**

- ...

Agenda

- Objects (review)
- Prototypes
- Delegation to prototypes
- **Classes**

Classes

- See **fraction5.js**, **fraction5client.js**

```
$ node fraction5client.js
Numerator 1: 1
Denominator 1: 2
Numerator 2: 3
Denominator 2: 4
f1: 1/2
f2: 3/4
f1 is not identical to f2
f1 is less than f2
-f1: -1/2
f1 + f2: 5/4
f1 - f2: -1/4
f1 * f2: 3/8
f1 / f2: 2/3
$
```

fraction5.js (Page 1 of 2)

```

1: //-----
2: // fraction5.js
3: // Author: Bob Dondero
4: //-----
5:
6: 'use strict';
7:
8: const euclid = require('./euclid.js');
9:
10: class Fraction {
11:   constructor(num=0, den=1) {
12:     if (arguments.length > 2)
13:       throw new Error('Too many arguments');
14:
15:     if (den === 0)
16:       throw new Error('Denominator cannot be zero');
17:
18:     this._num = num;
19:     this._den = den;
20:
21:     if (this._den < 0) {
22:       this._num *= -1;
23:       this._den *= -1;
24:     }
25:     if (this._num === 0)
26:       this._den = 1;
27:     else {
28:       let gcden = euclid.gcd(this._num, this._den);
29:       this._num /= gcden;
30:       this._den /= gcden;
31:     }
32:   }
33:
34:   toString() {
35:     return String(this._num) + '/' + String(this._den);
36:   }
37:
38:   compareTo(other) {
39:     if ((this._num * other._den) < (other._num * this._den))
40:       return -1;
41:     if ((this._num * other._den) > (other._num * this._den))
42:       return 1;
43:     return 0;
44:   }
45:
46:   negate() {
47:     return new Fraction(-this._num, this._den);
48:   }
49:
50:   add(other) {
51:     let newNum = (this._num * other._den) + (other._num * this._den);
52:     let newDen = this._den * other._den;
53:     return new Fraction(newNum, newDen);
54:   }
55:
56:   subtract(other) {
57:     let newNum = (this._num * other._den) - (other._num * this._den);
58:     let newDen = this._den * other._den;
59:     return new Fraction(newNum, newDen);
60:   }
61:
62:   multiply(other) {
63:     let newNum = this._num * other._num;
64:     let newDen = this._den * other._den;
65:     return new Fraction(newNum, newDen);

```

fraction5.js (Page 2 of 2)

```

66:   }
67:
68:   divide(other) {
69:     let newNum = this._num * other._den;
70:     let newDen = this._den * other._num;
71:     return new Fraction(newNum, newDen);
72:   }
73: }
74:
75: module.exports = { Fraction };

```

fraction5client.js (Page 1 of 2)

```

1: //-----
2: // fraction5client.js
3: // Author: Bob Dondero
4: //-----
5:
6: 'use strict';
7:
8: const readlineSync = require('readline-sync');
9: const fraction = require('./fraction5.js');
10:
11: //-----
12:
13: function readInt(prompt) {
14:   let line = readlineSync.question(prompt);
15:   if (line === '')
16:     throw new Error('Missing integer');
17:
18:   let n = Number(line);
19:   if (! Number.isInteger(n))
20:     throw new Error('Not an integer');
21:
22:   return n;
23: }
24:
25: //-----
26:
27: function main() {
28:   try {
29:     let n1 = readInt('Numerator 1: ');
30:     let d1 = readInt('Denominator 1: ');
31:     let n2 = readInt('Numerator 2: ');
32:     let d2 = readInt('Denominator 2: ');
33:
34:     let f1 = new fraction.Fraction(n1, d1);
35:     let f2 = new fraction.Fraction(n2, d2);
36:
37:     process.stdout.write('f1: ' + f1.toString() + '\n');
38:     process.stdout.write('f2: ' + String(f2) + '\n');
39:
40:     if (f1 === f2)
41:       process.stdout.write('f1 is identical to f2\n');
42:     else
43:       process.stdout.write('f1 is not identical to f2\n');
44:
45:     let compare = f1.compareTo(f2);
46:     if (compare < 0)
47:       process.stdout.write('f1 is less than f2\n');
48:     if (compare > 0)
49:       process.stdout.write('f1 is greater than f2\n');
50:     if (compare === 0)
51:       process.stdout.write('f1 is equal to f2\n');
52:
53:     let f3;
54:
55:     f3 = f1.negate();
56:     process.stdout.write('-f1: ' + String(f3) + '\n');
57:
58:     f3 = f1.add(f2);
59:     process.stdout.write('f1 + f2: ' + String(f3) + '\n');
60:
61:     f3 = f1.subtract(f2);
62:     process.stdout.write('f1 - f2: ' + String(f3) + '\n');
63:
64:     f3 = f1.multiply(f2);
65:     process.stdout.write('f1 * f2: ' + String(f3) + '\n');

```

fraction5client.js (Page 2 of 2)

```

66:
67:     f3 = f1.divide(f2);
68:     process.stdout.write('f1 / f2: ' + String(f3) + '\n');
69:   }
70:   catch (e) {
71:     process.stderr.write(e + '\n');
72:   }
73: }
74:
75: if (require.main === module)
76:   main();

```

Classes

- JavaScript really doesn't have:
 - Classes
 - Objects as instances of classes
- JavaScript has:
 - Objects
 - Delegation to prototypes

Aside: Prototype Chains

- JavaScript really doesn't have:
 - Inheritance
- JavaScript has:
 - Prototype chains
 - (Beyond our scope)

Aside: this

- **Question:** How is `this` bound within a function `f()`?
- **Answer:** Depends upon how `f()` is called:

Function Call	Binding of <code>this</code>
<code>f()</code>	In <code>f()</code> , <code>this</code> is undefined
<code>obj.f()</code>	In <code>f()</code> , <code>this</code> is bound to <code>obj</code>
<code>new f()</code>	In <code>f()</code> , <code>this</code> is bound to a new empty object

JavaScript Commentary

- **Classes** evolutionary path
 - Simula, Smalltalk
 - C++, Java, Python, ...
- **Delegation to prototypes** evolutionary path
 - Self
 - JavaScript, TypeScript

Summary

- We have covered:
 - Prototypes
 - Delegation to prototypes
 - Classes