The JavaScript Language (Part 3)

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)

```javascript
//-----------------------------------------------------------------------
// fraction2.js
// Author: Bob Dondero
//-----------------------------------------------------------------------

'use strict';

const euclid = require('./euclid.js');

function createFraction(num=0, den=1)
{
   if (arguments.length > 2)
      throw new Error('Too many arguments');

   if (den === 0)
      throw new Error('Denominator cannot be zero');

   let f = {};

   f._num = num;
   f._den = den;

   if (f._den < 0) {
      f._num *= -1;
      f._den *= -1;
   }
   if (f._num === 0)
      f._den = 1;
   else {
      let gcden = euclid.gcd(f._num, f._den);
      f._num /= gcden;
      f._den /= gcden;
   }

   f.toString = function() {
      return String(this._num) + '/' + String(this._den);
   };

   f.compareTo = function(other) {
      if ((this._num * other._den) < (other._num * this._den))
         return -1;
      if ((this._num * other._den) > (other._num * this._den))
         return 1;
      return 0;
   };

   f.negate = function() {
      return createFraction(-this._num, this._den);
   };

   f.add = function(other) {
      let newNum = (this._num * other._den) + (other._num * this._den);
      let newDen = this._den * other._den;
      return createFraction(newNum, newDen);
   };

   f.subtract = function(other) {
      let newNum = (this._num * other._den) - (other._num * this._den);
      let newDen = this._den * other._den;
      return createFraction(newNum, newDen);
   };

   f.multiply = function(other) {
      let newNum = this._num * other._num;
      let newDen = this._den * other._den;
```

fraction2.js (Page 2 of 2)

```javascript
      return createFraction(newNum, newDen);
   };

   f.divide = function(other) {
      let newNum = this._num * other._den;
      let newDen = this._den * other._num;
      return createFraction(newNum, newDen);
   };

   return f;
}

module.exports = { createFraction };
```

fraction2client.js (Page 1 of 2)

```javascript
//-----------------------------------------------------------------------
// fraction2client.js
// Author: Bob Dondero
//-----------------------------------------------------------------------

'use strict';

const readlineSync = require('readline-sync');
const fraction = require('./fraction2.js');

//-----------------------------------------------------------------------

function readInt(prompt) {
   let line = readlineSync.question(prompt);
   if (line === '')
      throw new Error('Missing integer');
   if (isNaN(line))
      throw new Error('Not a number');
   let n = Number(line);
   if (! Number.isInteger(n))
      throw new Error('Not an integer');
   return n;
}

//-----------------------------------------------------------------------

function main() {
   try {
      let n1 = readInt('Numerator 1: ');
      let d1 = readInt('Denominator 1: ');
      let n2 = readInt('Numerator 2: ');
      let d2 = readInt('Denominator 2: ');

      let f1 = fraction.createFraction(n1, d1);
      let f2 = fraction.createFraction(n2, d2);

      process.stdout.write('f1: ' + f1.toString() + '\n');
      process.stdout.write('f2: ' + String(f2) + '\n');

      if (f1 === f2)
         process.stdout.write('f1 is identical to f2\n');
      else
         process.stdout.write('f1 is not identical to f2\n');

      let compare = f1.compareTo(f2);
      if (compare < 0)
         process.stdout.write('f1 is less than f2\n');
      if (compare > 0)
         process.stdout.write('f1 is greater than f2\n');
      if (compare === 0)
         process.stdout.write('f1 is equal to f2\n');

      let f3;

      f3 = f1.negate();
      process.stdout.write('-f1: ' + String(f3) + '\n');

      f3 = f1.add(f2);
      process.stdout.write('f1 + f2: ' + String(f3) + '\n');

      f3 = f1.subtract(f2);
      process.stdout.write('f1 - f2: ' + String(f3) + '\n');

      f3 = f1.multiply(f2);
      process.stdout.write('f1 * f2: ' + String(f3) + '\n');
```

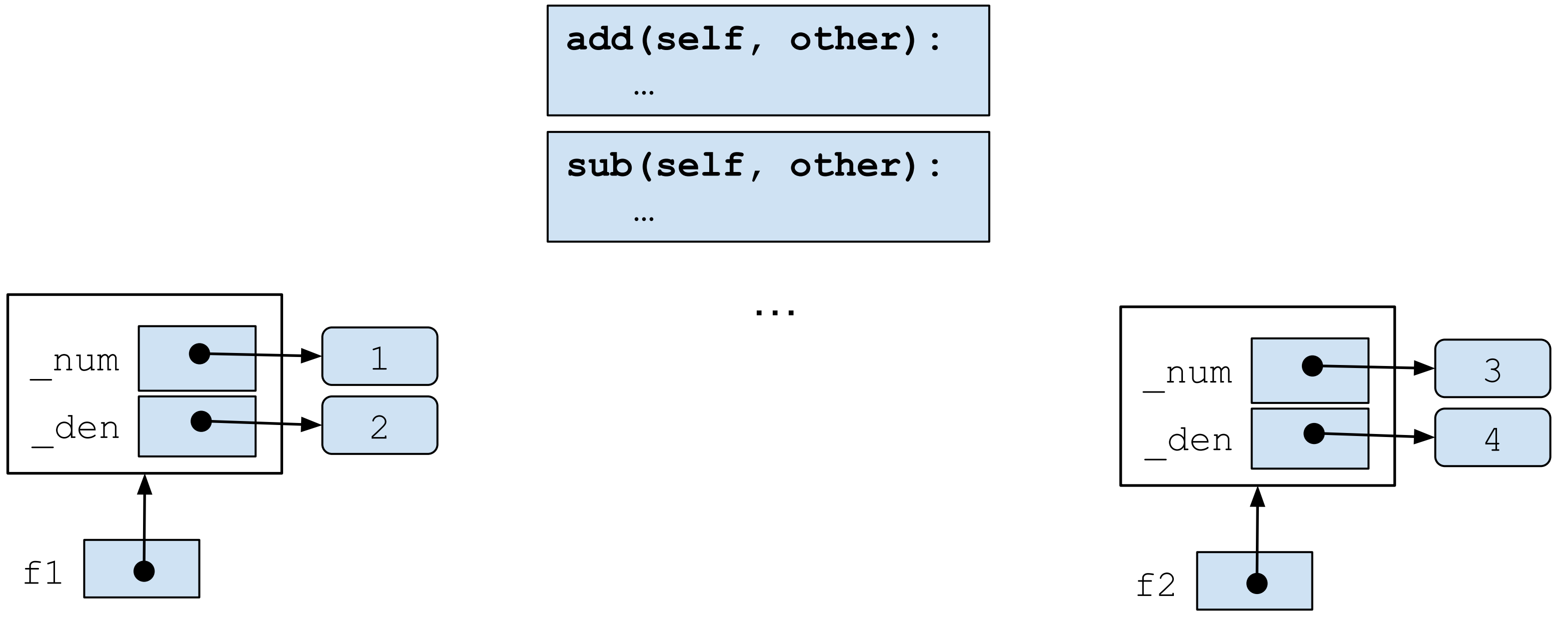
fraction2client.js (Page 2 of 2)

```javascript

      f3 = f1.divide(f2);
      process.stdout.write('f1 / f2: ' + String(f3) + '\n');
   }
   catch (e) {
      process.stderr.write(String(e) + '\n');
   }
}

if (require.main === module)
   main();
```

Objects (Review)

In Python

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

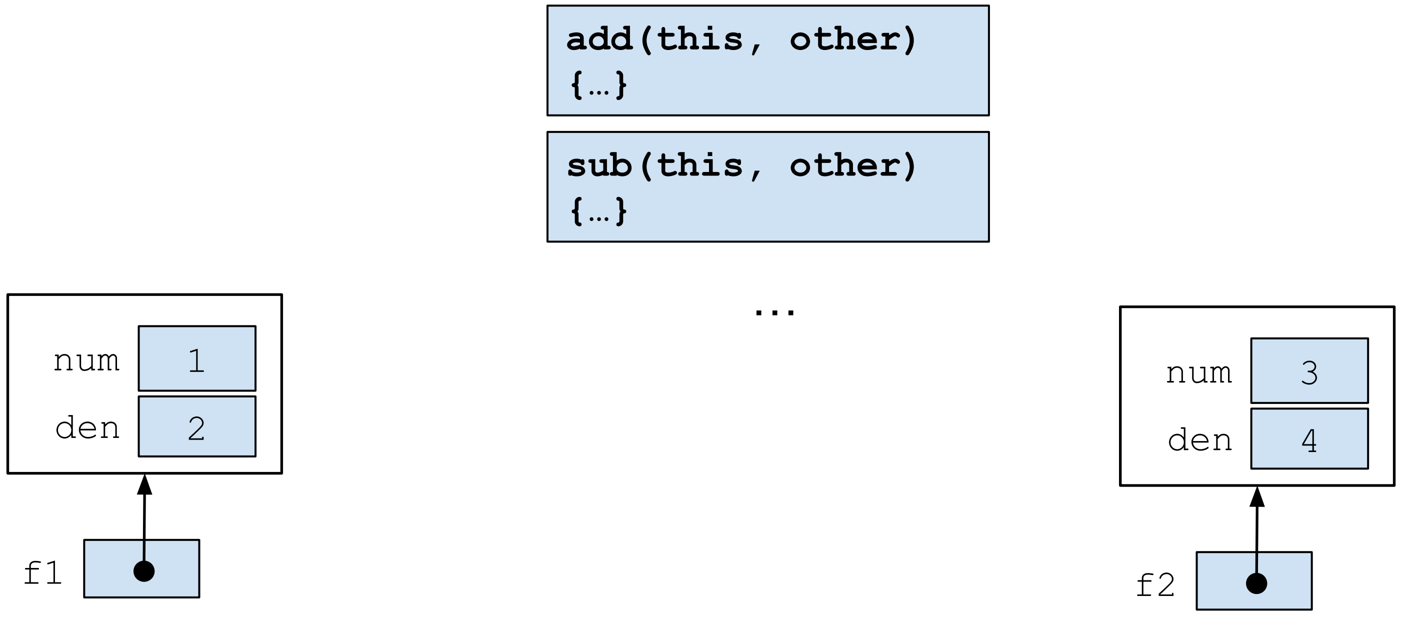


Explicit `self` parameter allows `Fraction` objects to share same function defs

Objects (Review)

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

In Java

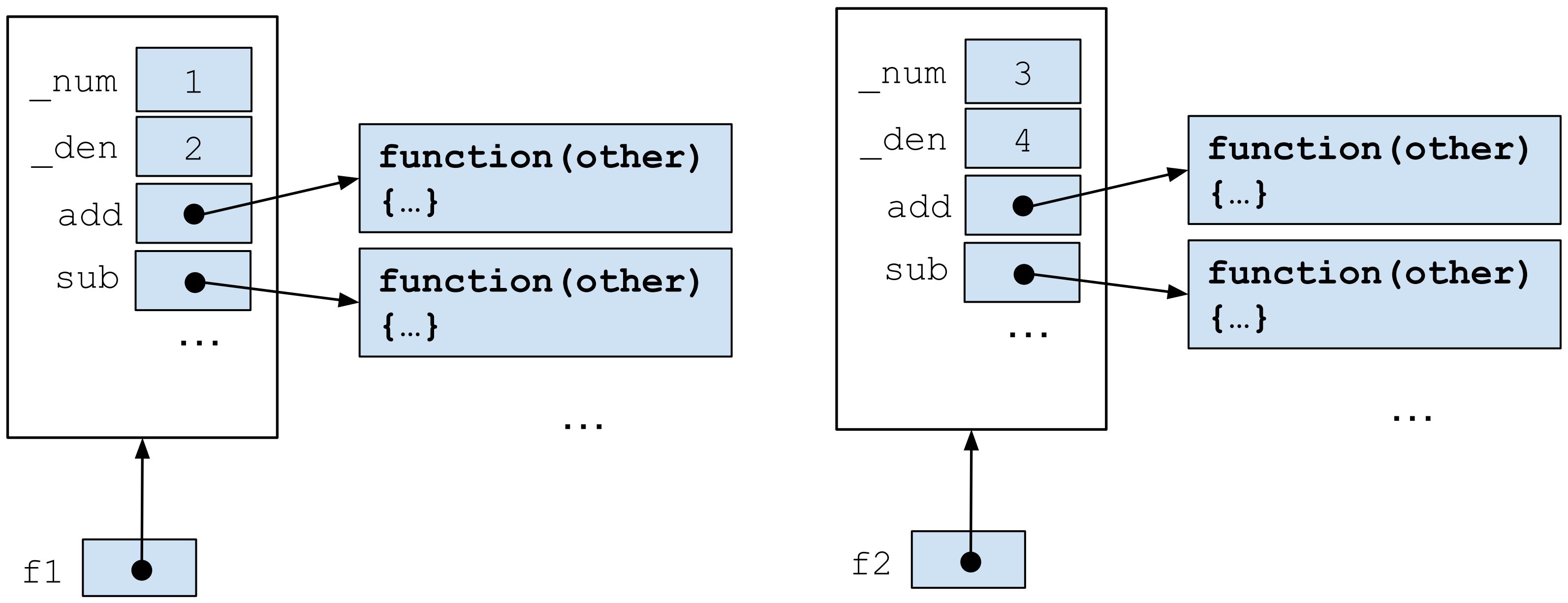


Implicit `this` parameter allows `Fraction` objects to share same method defs

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)

```
//-----------------------------------------------------------------------
// fraction3.js
// Author: Bob Dondero
//-----------------------------------------------------------------------

'use strict';

const euclid = require('./euclid.js');

function Fraction(num=0, den=1) {
   if (arguments.length > 2)
      throw new Error('Too many arguments');

   if (den === 0)
      throw new Error('Denominator cannot be zero');

   this._num = num;
   this._den = den;

   if (this._den < 0) {
      this._num *= -1;
      this._den *= -1;
   }
   if (this._num === 0)
      this._den = 1;
   else {
      let gcden = euclid.gcd(this._num, this._den);
      this._num /= gcden;
      this._den /= gcden;
   }

   this.toString = function() {
      return String(this._num) + '/' + String(this._den);
   };

   this.compareTo = function(other) {
      if ((this._num * other._den) < (other._num * this._den))
         return -1;
      if ((this._num * other._den) > (other._num * this._den))
         return 1;
      return 0;
   };

   this.negate = function() {
      return new Fraction(-this._num, this._den);
   };

   this.add = function(other) {
      let newNum = (this._num * other._den) + (other._num * this._den);
      let newDen = this._den * other._den;
      return new Fraction(newNum, newDen);
   };

   this.subtract = function(other) {
      let newNum = (this._num * other._den) - (other._num * this._den);
      let newDen = this._den * other._den;
      return new Fraction(newNum, newDen);
   };

   this.multiply = function(other) {
      let newNum = this._num * other._num;
      let newDen = this._den * other._den;
      return new Fraction(newNum, newDen);
   };

```

fraction3.js (Page 2 of 2)

```
   this.divide = function(other) {
      let newNum = this._num * other._den;
      let newDen = this._den * other._num;
      return new Fraction(newNum, newDen);
   };
}

module.exports = { Fraction };
```

fraction3client.js (Page 1 of 2)

```
//-----------------------------------------------------------------------
// fraction3client.js
// Author: Bob Dondero
//-----------------------------------------------------------------------

'use strict';

const readlineSync = require('readline-sync');
const fraction = require('./fraction3.js');

//-----------------------------------------------------------------------

function readInt(prompt) {
   let line = readlineSync.question(prompt);
   if (line === '')
      throw new Error('Missing integer');

   let n = Number(line);
   if (! Number.isInteger(n))
      throw new Error('Not an integer');

   return n;
}

//-----------------------------------------------------------------------

function main() {
   try {
      let n1 = readInt('Numerator 1: ');
      let d1 = readInt('Denominator 1: ');
      let n2 = readInt('Numerator 2: ');
      let d2 = readInt('Denominator 2: ');

      let f1 = new fraction.Fraction(n1, d1);
      let f2 = new fraction.Fraction(n2, d2);

      process.stdout.write('f1: ' + f1.toString() + '\n');
      process.stdout.write('f2: ' + String(f2) + '\n');

      if (f1 === f2)
         process.stdout.write('f1 is identical to f2\n');
      else
         process.stdout.write('f1 is not identical to f2\n');

      let compare = f1.compareTo(f2);
      if (compare < 0)
         process.stdout.write('f1 is less than f2\n');
      if (compare > 0)
         process.stdout.write('f1 is greater than f2\n');
      if (compare === 0)
         process.stdout.write('f1 is equal to f2\n');

      let f3;

      f3 = f1.negate();
      process.stdout.write('-f1: ' + String(f3) + '\n');

      f3 = f1.add(f2);
      process.stdout.write('f1 + f2: ' + String(f3) + '\n');

      f3 = f1.subtract(f2);
      process.stdout.write('f1 - f2: ' + String(f3) + '\n');

      f3 = f1.multiply(f2);
      process.stdout.write('f1 * f2: ' + String(f3) + '\n');
```

fraction3client.js (Page 2 of 2)

```

      f3 = f1.divide(f2);
      process.stdout.write('f1 / f2: ' + String(f3) + '\n');
   }
   catch (e) {
      process.stderr.write(e + '\n');
   }
}

if (require.main === module)
   main();
```

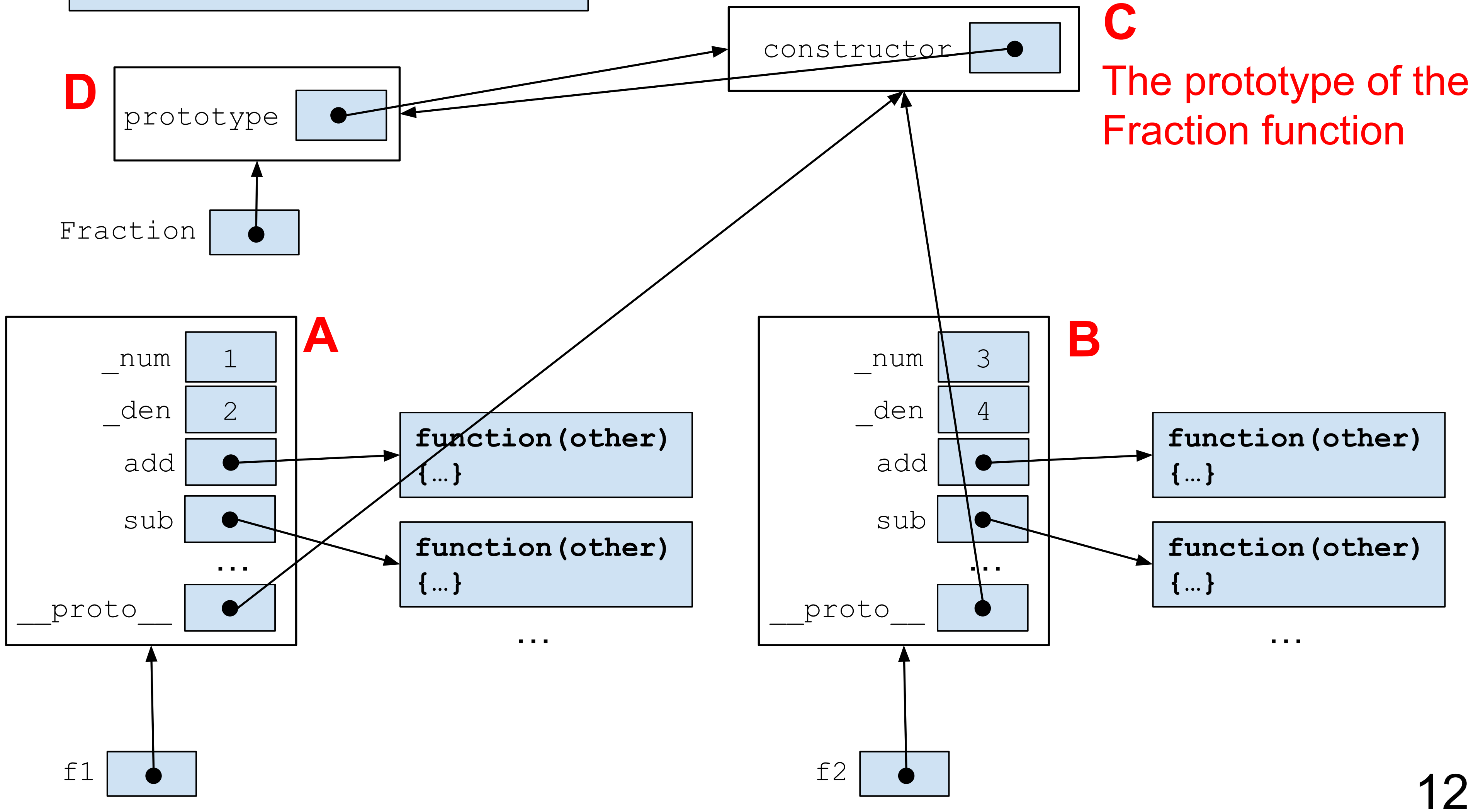
Prototypes

- **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)

```
//-----------------------------------------------------------------------
// fraction4.js
// Author: Bob Dondero
//-----------------------------------------------------------------------

'use strict';

const euclid = require('./euclid.js');

function Fraction(num=0, den=1) {
   if (arguments.length > 2)
      throw new Error('Too many arguments');

   if (den === 0)
      throw new Error('Denominator cannot be zero');

   this._num = num;
   this._den = den;

   if (this._den < 0) {
      this._num *= -1;
      this._den *= -1;
   }
   if (this._num === 0)
      this._den = 1;
   else {
      let gcden = euclid.gcd(this._num, this._den);
      this._num /= gcden;
      this._den /= gcden;
   }
}

Fraction.prototype.toString = function() {
   return String(this._num) + '/' + String(this._den);
};

Fraction.prototype.compareTo = function(other) {
   if ((this._num * other._den) < (other._num * this._den)) return -1;
   if ((this._num * other._den) > (other._num * this._den)) return 1;
   return 0;
};

Fraction.prototype.negate = function() {
   return new Fraction(-this._num, this._den);
};

Fraction.prototype.add = function(other) {
   let newNum = (this._num * other._den) + (other._num * this._den);
   let newDen = this._den * other._den;
   return new Fraction(newNum, newDen);
};

Fraction.prototype.subtract = function(other) {
   let newNum = (this._num * other._den) - (other._num * this._den);
   let newDen = this._den * other._den;
   return new Fraction(newNum, newDen);
};

Fraction.prototype.multiply = function(other) {
   let newNum = this._num * other._num;
   let newDen = this._den * other._den;
   return new Fraction(newNum, newDen);
};

Fraction.prototype.divide = function(other) {
```

fraction4.js (Page 2 of 2)

```
   let newNum = this._num * other._den;
   let newDen = this._den * other._num;
   return new Fraction(newNum, newDen);
};

module.exports = { Fraction };
```

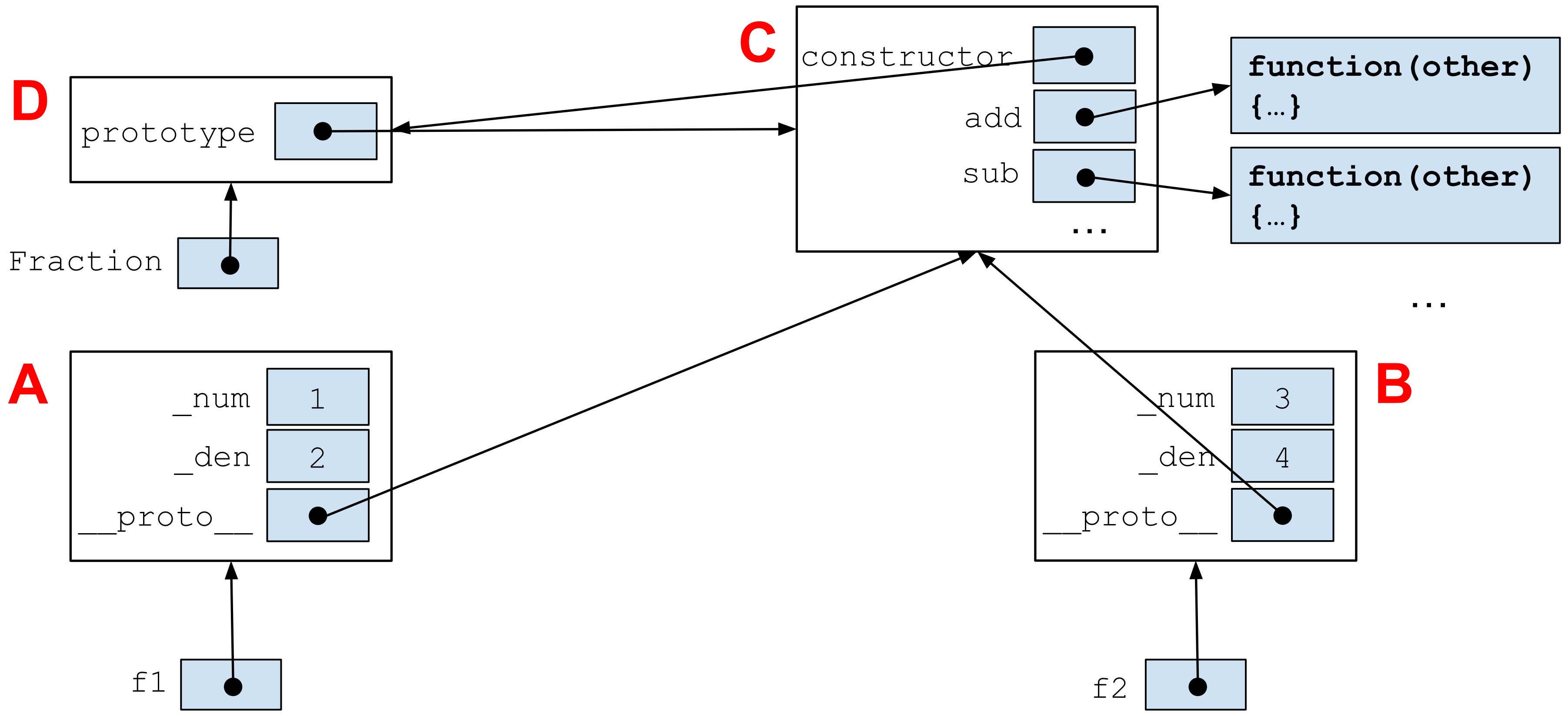
fraction4client.js (Page 1 of 2)

```
//-----------------------------------------------------------------------
// fraction4client.js
// Author: Bob Dondero
//-----------------------------------------------------------------------

'use strict';

const readline = require('readline-sync');
const fraction = require('./fraction4.js');

//-----------------------------------------------------------------------

function readInt(prompt) {
   let line = readline.question(prompt);
   if (line === '')
      throw new Error('Missing integer');

   let n = Number(line);
   if (! Number.isInteger(n))
      throw new Error('Not an integer');

   return n;
}

//-----------------------------------------------------------------------

function main() {
   try    {
      let n1 = readInt('Numerator 1: ');
      let d1 = readInt('Denominator 1: ');
      let n2 = readInt('Numerator 2: ');
      let d2 = readInt('Denominator 2: ');

      let f1 = new fraction.Fraction(n1, d1);
      let f2 = new fraction.Fraction(n2, d2);

      process.stdout.write('f1: ' + f1.toString() + '\n');
      process.stdout.write('f2: ' + String(f2) + '\n');
      process.stdout.write('f2: ' + f2 + '\n');

      if (f1 === f2)
         process.stdout.write('f1 is identical to f2\n');
      else
         process.stdout.write('f1 is not identical to f2\n');

      let compare = f1.compareTo(f2);
      if (compare < 0)
         process.stdout.write('f1 is less than f2\n');
      if (compare > 0)
         process.stdout.write('f1 is greater than f2\n');
      if (compare === 0)
         process.stdout.write('f1 is equal to f2\n');

      let f3;

      f3 = f1.negate();
      process.stdout.write('-f1: ' + String(f3) + '\n');

      f3 = f1.add(f2);
      process.stdout.write('f1 + f2: ' + String(f3) + '\n');

      f3 = f1.subtract(f2);
      process.stdout.write('f1 - f2: ' + String(f3) + '\n');

      f3 = f1.multiply(f2);
```

fraction4client.js (Page 2 of 2)

```
      process.stdout.write('f1 * f2: ' + String(f3) + '\n');

      f3 = f1.divide(f2);
      process.stdout.write('f1 / f2: ' + String(f3) + '\n');
   }
   catch (e) {
      process.stderr.write(e + '\n');
   }
}

if (require.main === module)
   main();
```

Delegation to Prototypes



`f1.add(f2)` => runtime looks for `f1.add()` then `f1.__proto__.add()`
`f2.add(f1)` => 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)

```
//-----------------------------------------------------------------------
// fraction5.js
// Author: Bob Dondero
//-----------------------------------------------------------------------

'use strict';

const euclid = require('./euclid.js');

class Fraction {
   constructor(num=0, den=1) {
      if (arguments.length > 2)
         throw new Error('Too many arguments');

      if (den === 0)
         throw new Error('Denominator cannot be zero');

      this._num = num;
      this._den = den;

      if (this._den < 0) {
         this._num *= -1;
         this._den *= -1;
      }
      if (this._num === 0)
         this._den = 1;
      else {
         let gcden = euclid.gcd(this._num, this._den);
         this._num /= gcden;
         this._den /= gcden;
      }
   }

   toString() {
      return String(this._num) + '/' + String(this._den);
   }

   compareTo(other) {
      if ((this._num * other._den) < (other._num * this._den))
         return -1;
      if ((this._num * other._den) > (other._num * this._den))
         return 1;
      return 0;
   }

   negate() {
      return new Fraction(-this._num, this._den);
   }

   add(other) {
      let newNum = (this._num * other._den) + (other._num * this._den);
      let newDen = this._den * other._den;
      return new Fraction(newNum, newDen);
   }

   subtract(other) {
      let newNum = (this._num * other._den) - (other._num * this._den);
      let newDen = this._den * other._den;
      return new Fraction(newNum, newDen);
   }

   multiply(other) {
      let newNum = this._num * other._num;
      let newDen = this._den * other._den;
      return new Fraction(newNum, newDen);
```

fraction5.js (Page 2 of 2)

```
   }

   divide(other) {
      let newNum = this._num * other._den;
      let newDen = this._den * other._num;
      return new Fraction(newNum, newDen);
   }
}

module.exports = { Fraction };
```

fraction5client.js (Page 1 of 2)

```
//-----------------------------------------------------------------------
// fraction5client.js
// Author: Bob Dondero
//-----------------------------------------------------------------------

'use strict';

const readlineSync = require('readline-sync');
const fraction = require('./fraction5.js');

//-----------------------------------------------------------------------

function readInt(prompt) {
   let line = readlineSync.question(prompt);
   if (line === '')
      throw new Error('Missing integer');

   let n = Number(line);
   if (! Number.isInteger(n))
      throw new Error('Not an integer');

   return n;
}

//-----------------------------------------------------------------------

function main() {
   try {
      let n1 = readInt('Numerator 1: ');
      let d1 = readInt('Denominator 1: ');
      let n2 = readInt('Numerator 2: ');
      let d2 = readInt('Denominator 2: ');

      let f1 = new fraction.Fraction(n1, d1);
      let f2 = new fraction.Fraction(n2, d2);

      process.stdout.write('f1: ' + f1.toString() + '\n');
      process.stdout.write('f2: ' + String(f2) + '\n');

      if (f1 === f2)
         process.stdout.write('f1 is identical to f2\n');
      else
         process.stdout.write('f1 is not identical to f2\n');

      let compare = f1.compareTo(f2);
      if (compare < 0)
         process.stdout.write('f1 is less than f2\n');
      if (compare > 0)
         process.stdout.write('f1 is greater than f2\n');
      if (compare === 0)
         process.stdout.write('f1 is equal to f2\n');

      let f3;

      f3 = f1.negate();
      process.stdout.write('-f1: ' + String(f3) + '\n');

      f3 = f1.add(f2);
      process.stdout.write('f1 + f2: ' + String(f3) + '\n');

      f3 = f1.subtract(f2);
      process.stdout.write('f1 - f2: ' + String(f3) + '\n');

      f3 = f1.multiply(f2);
      process.stdout.write('f1 * f2: ' + String(f3) + '\n');
```

fraction5client.js (Page 2 of 2)

```

      f3 = f1.divide(f2);
      process.stdout.write('f1 / f2: ' + String(f3) + '\n');
   }
   catch (e) {
      process.stderr.write(e + '\n');
   }
}

if (require.main === module)
   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 `this`
`f()`	In `f()`, `this` is `undefined`
`obj.f()`	In `f()`, `this` is bound to `obj`
`new f()`	In `f()`, `this` 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