

The JavaScript Language (Part 2)

Copyright © 2022 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

- **Modules**
- Objects

Modules

- See **euclid.js** and **euclidclient2.js**
 - The job:
 - Same as euclidclient1.js, but...
 - Factors program into modules

```
$ node euclidclient2.js
Enter the first integer:
8
Enter the second integer:
12
gcd: 4
lcm: 24
$
```

4

Modules

[see slide]

Code notes

Multi-file programs via Node.js modules

Module must explicitly export names to be used by its clients

Generalizing...

Modules

- Kinds of JavaScript modules
 - **Node.js** modules
 - Used by Node.js
 - Not used by browsers
 - **ES6** modules
 - Used in (recent) browsers
 - Not used by Node.js
 - (We'll see when studying React)

Agenda

- Modules
- **Objects**

Fraction Examples

- Fraction examples
 - Multiple versions
 - All have (nearly) identical behavior
 - Each uses additional JavaScript features
 - Generally, each is better than the previous

Objects

- See **fraction1.js**, **fraction1client.js**
 - The job:
 - Manipulate fraction objects

Objects

- See [fraction1.js](#), [fraction1client.js](#)

```
$ node fraction1client.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
$
```

9

Objects

[see slide]

Code notes

Creating an object:

```
let obj1 = {};
```

```
let obj2 =
```

```
{prop1: value1, prop2: value2, ...};
```

Accessing property values

```
let value1 = obj2.prop1;
```

Adding properties dynamically

```
obj2.prop3 = value3;
```

Objects

- **Problem**

- Instead of calling functions:
 - `f3 = fraction.add(f1, f2);`
- We want to send messages:
 - `f3 = f1.add(f2);`

- **Solution**

- The value of an object property can be a function definition...

Objects

- See **fraction2.js**, **fraction2client.js**
 - The job:
 - Same as fraction1.js and fraction1client.js

11

Objects

Code notes

Value of an object property can be a function definition

```
obj.prop = function(...) {...};
```

To call a function that is the value of an object property

```
obj.prop(...);
```

Within the function, this is bound to obj

Objects

- **Problem:**
 - Space inefficiency...

Objects

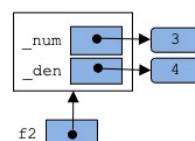
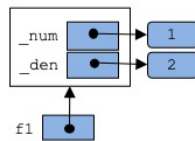
Python

```
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

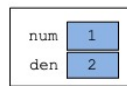
Java

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

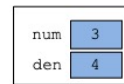
```
add(this, other)  
{...}
```

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

...



f1



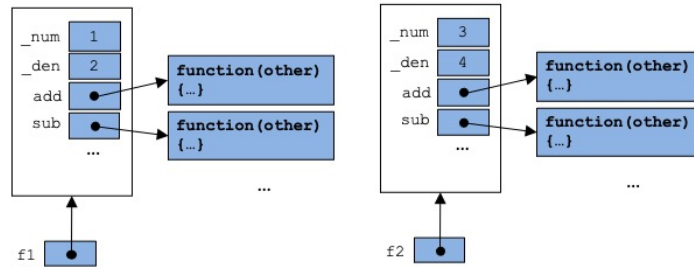
f2

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

Objects

JavaScript (so far)

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



Each fraction object has its own set of function defs
Many fraction objects => massive space inefficiency

Summary

- We have covered:
 - Modules
 - Objects