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)

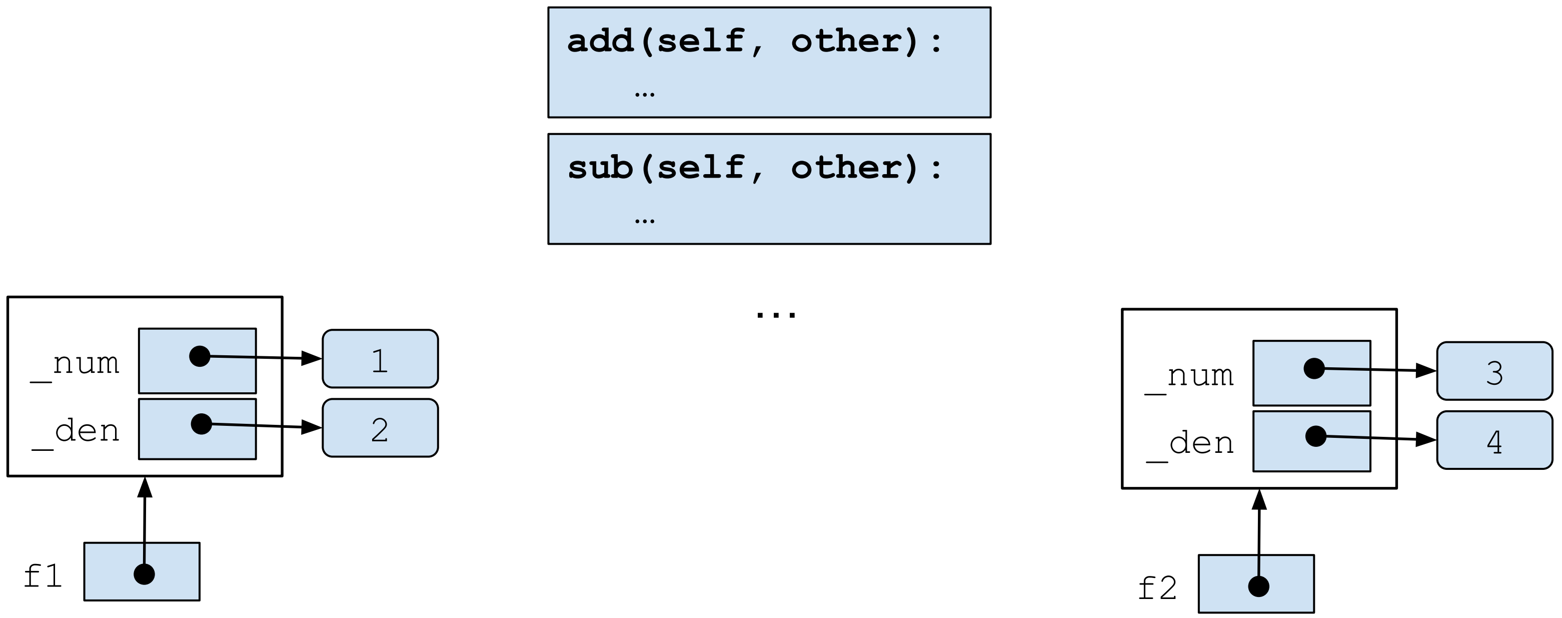
- Recall **fraction2.js**, **fraction2client.js**…

```
$ 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
$
```

Objects (Review)

In Python

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

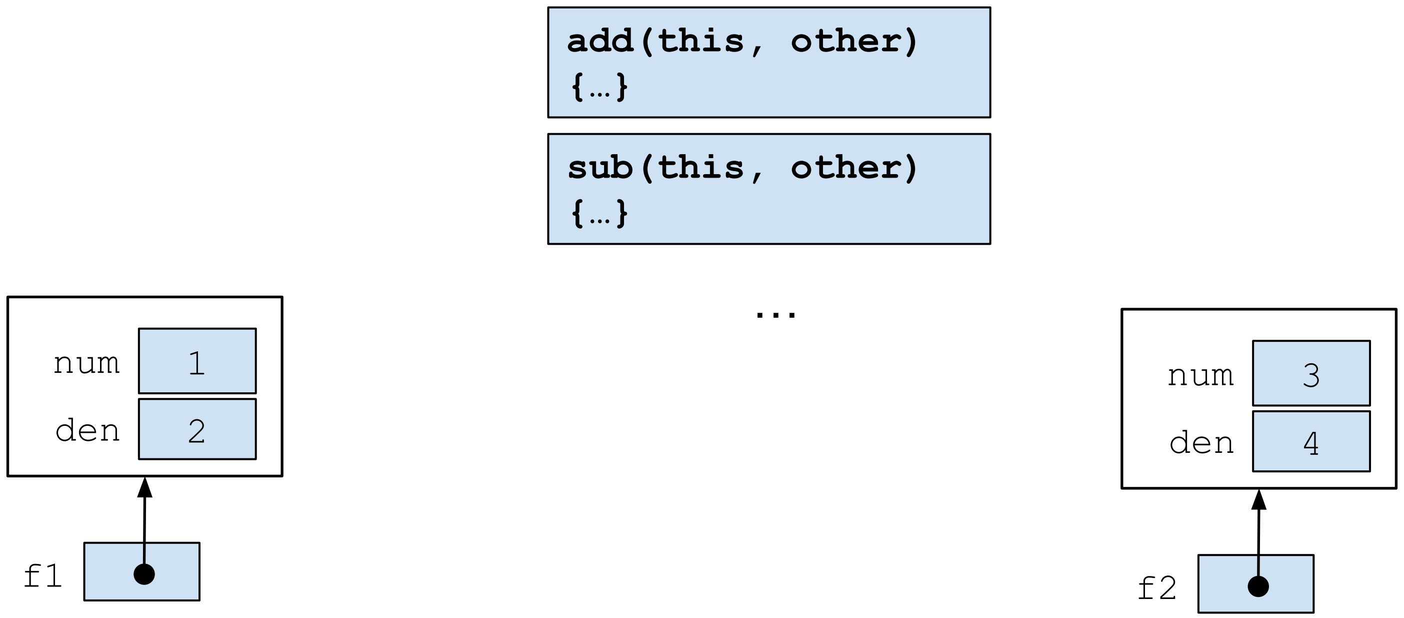


Explicit `self` parameter allows `Fraction` objects to share same function defs

Objects (Review)

In Java

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

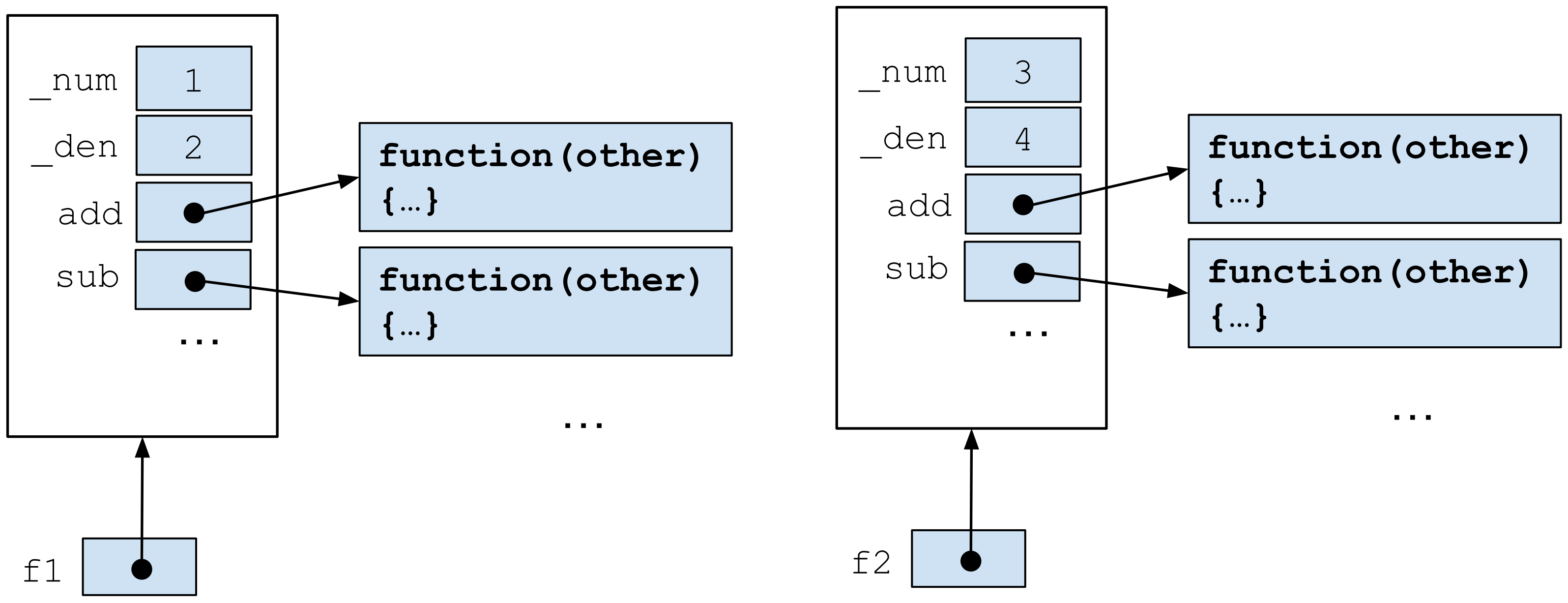


Implicit `this` parameter allows `Fraction` objects to share same method defs

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
$
```

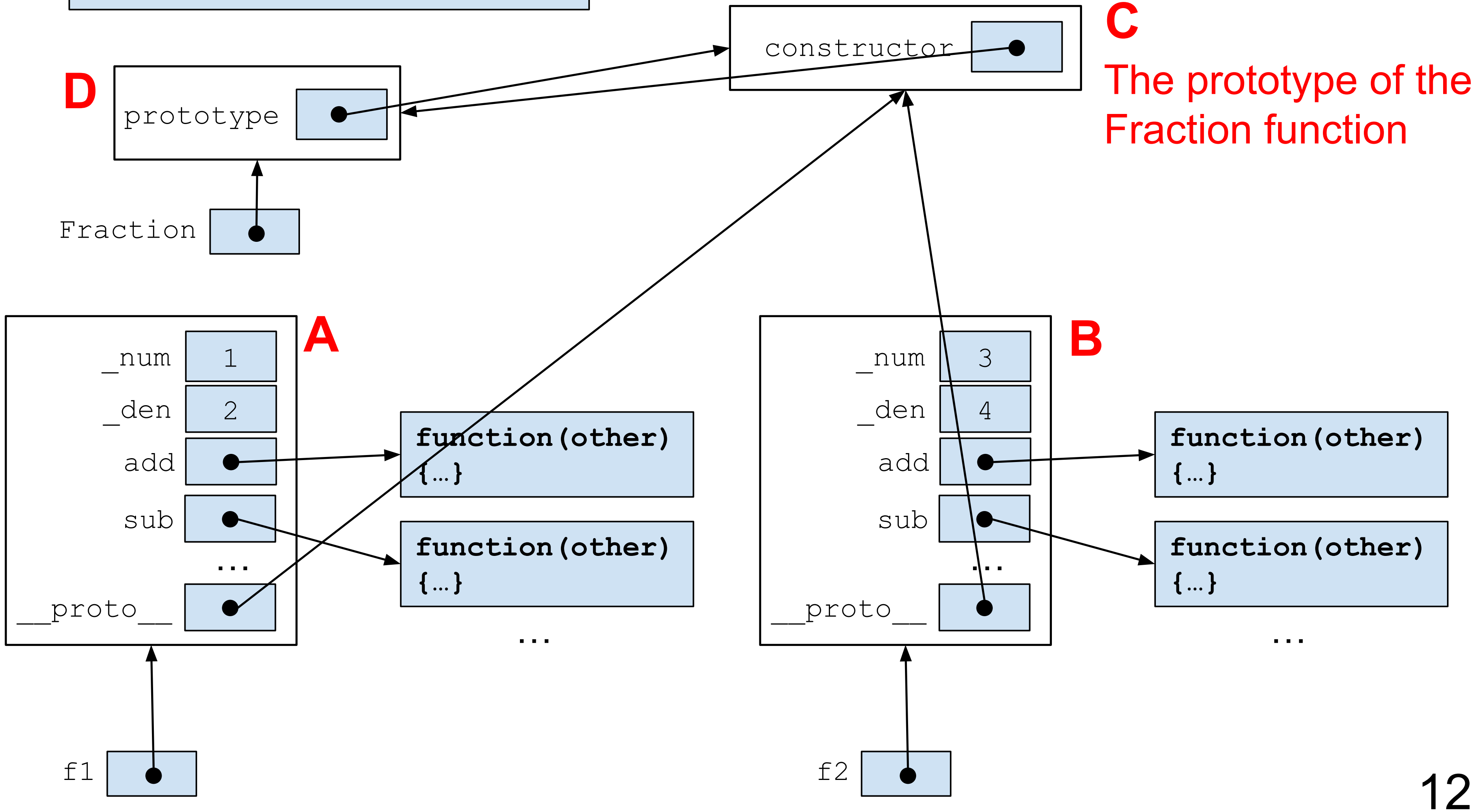
Prototypes

- **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

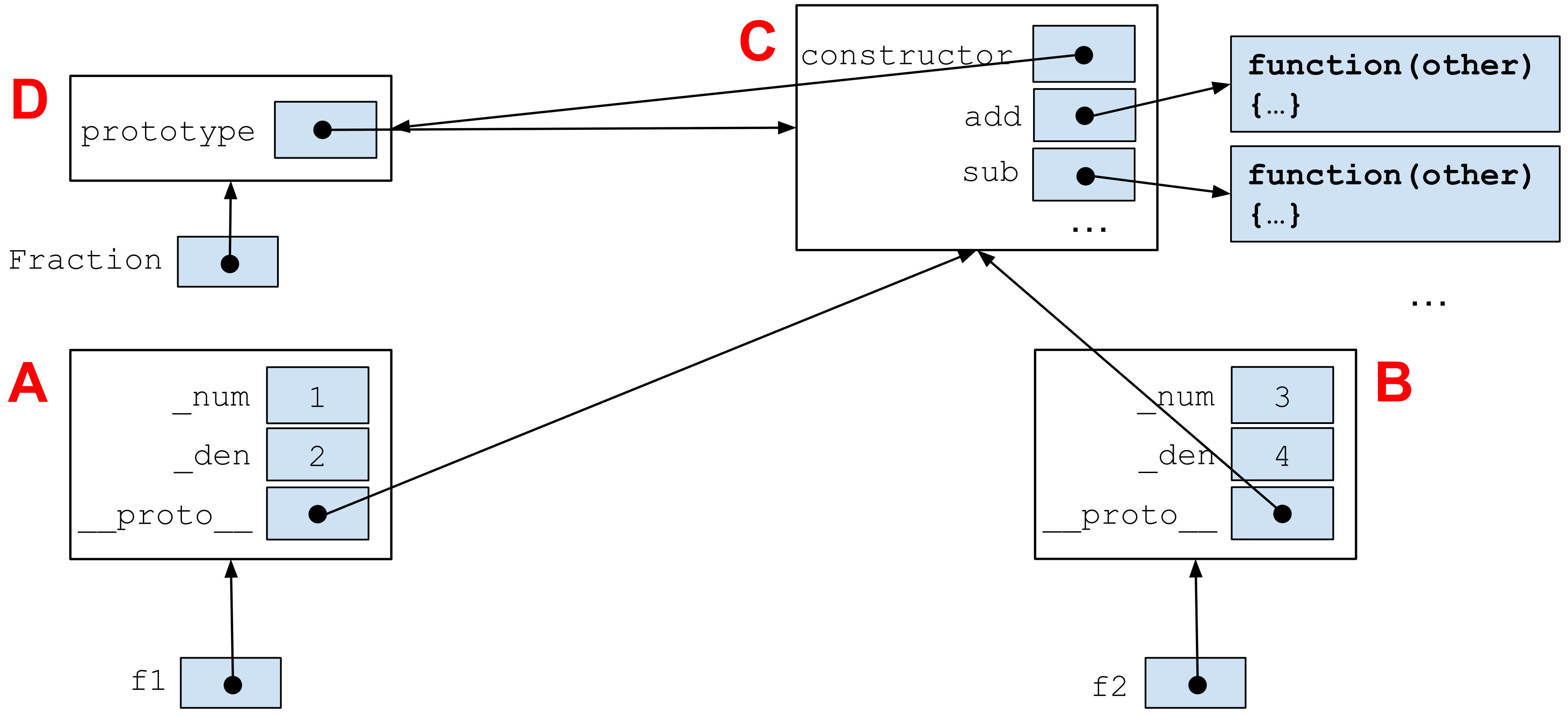
- Objects (review)
- Prototypes
- **Delegation to prototypes**
- Classes

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
$
```

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
$
```

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