International Workshop on Smalltalk Technologies Lyon, France; August 29th-31st, 2023

PharJS

Transpiling Pharo Classes to JS ECMAScript 5 versus ECMAScript 6

Noury Bouraqadi & Dave Mason

IMT Nord Europe
École Mines-Télécom
IMT-Université de Lille

Toronto Metropolitan University

Pharo 100%

Development

Production

100% Javascript

Pharo 100%

D

ECMAScript
5

Production

100% Javascript

Pharo 100%

ECMAScript
6?

D roduction

ECMAScript
5

100% Javascript

EcmaScript 5 vs EcmaScript 6

- Prototypes Only
- Dynamic Object Structure
- Whitebox Objects
- Reified Functions
- *this* Pseudo-variable
- Constructor Functions
- *new* Operator

ES5

+

Class Related Constructs

- Classes Definition
- Class Inheritance
- Instance Methods
- Class Methods (static)
- *super* Pseudo-variable

Class Transpilation by Example

Counter

count
defaultInstance

initialize
increment
createDefaultInstance
getDefaultInstance
resetDefaultInstance

```
initialize
    super initialize.
    count := 0
```

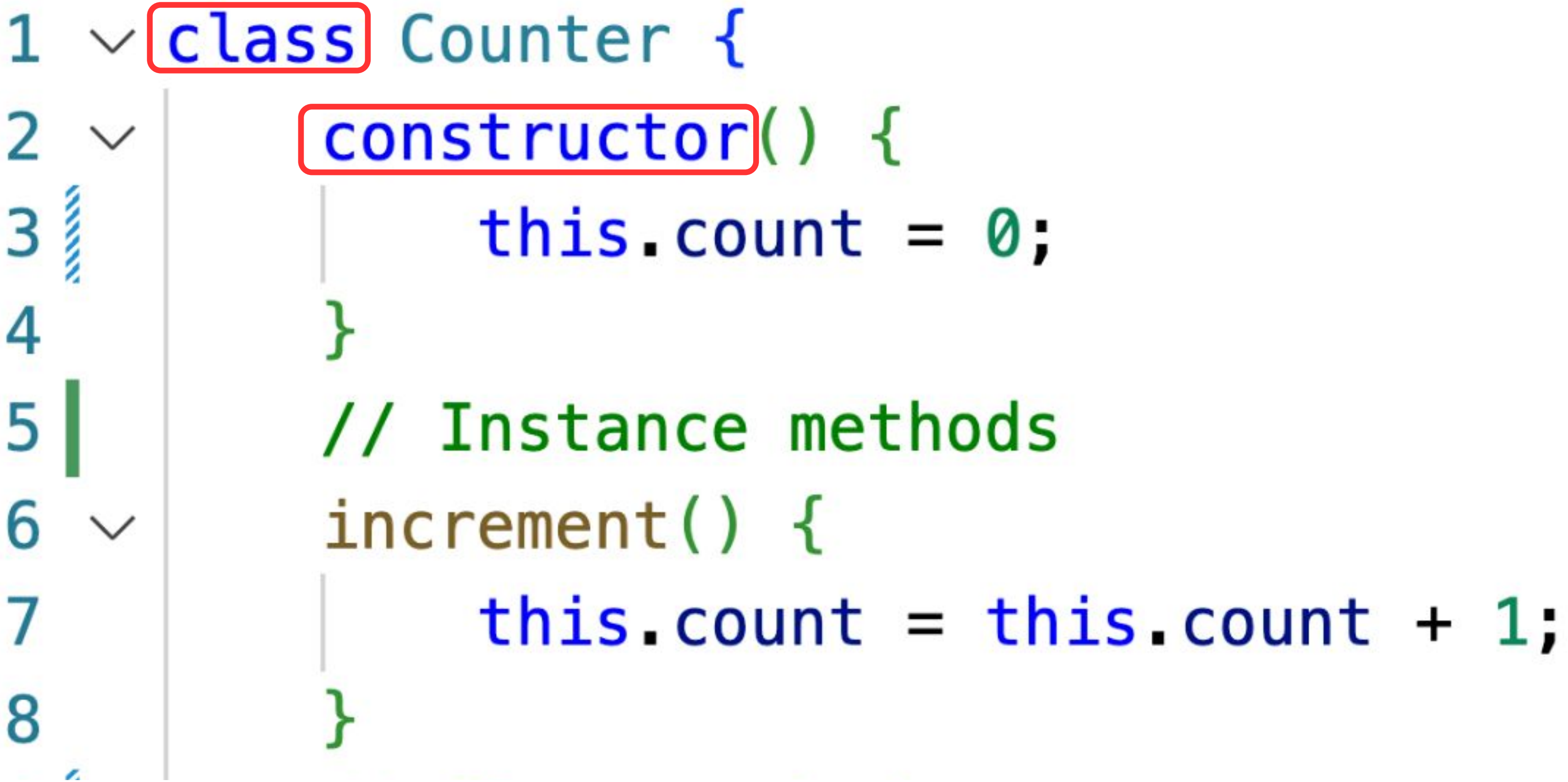
```
increment
    count := count + 1
```

```
createDefaultInstance
    ^defaultInstance  := self new
```

```
getDefaultInstance
    ^defaultInstance ifNil: [
        self createDefaultInstance]
```

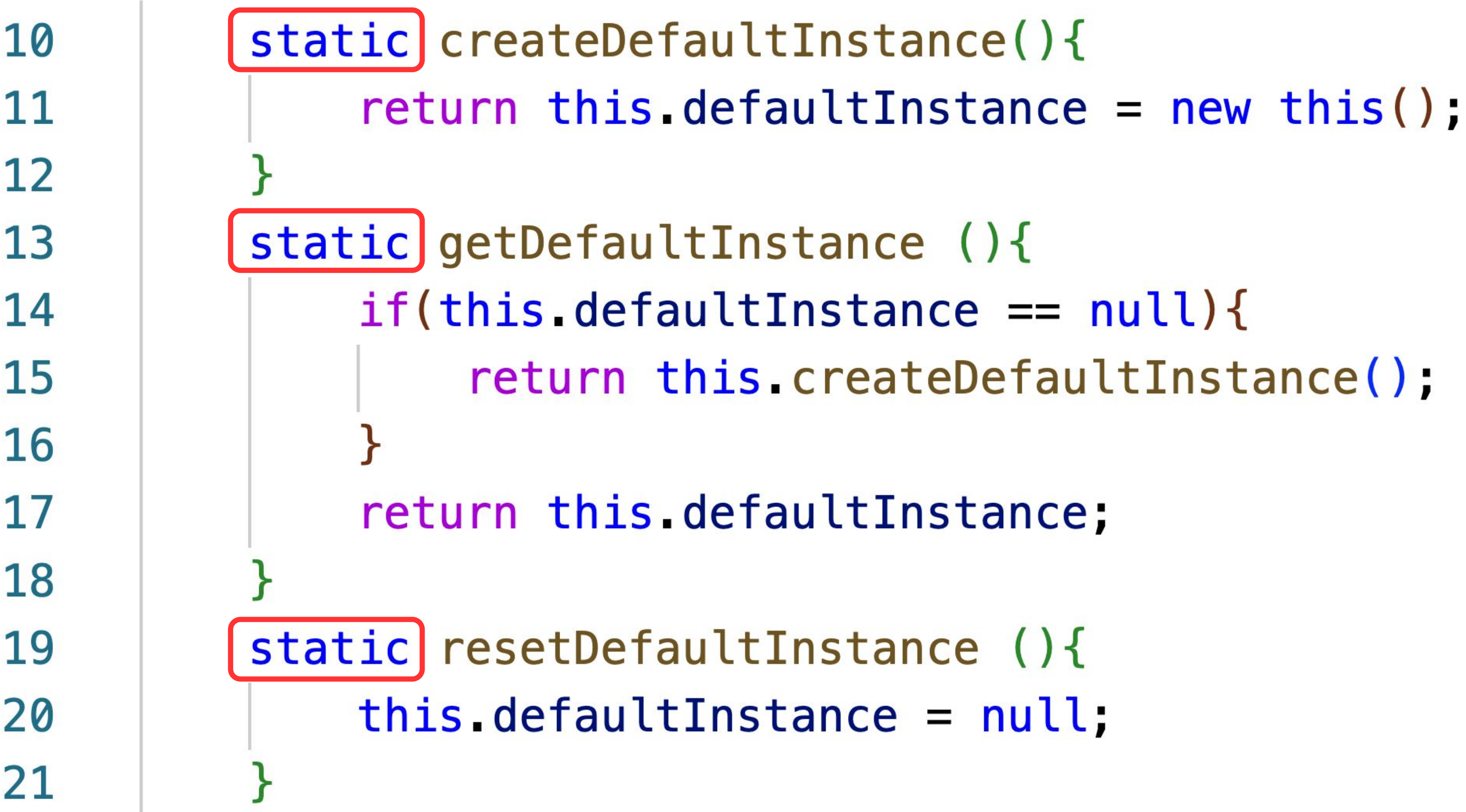
```
resetDefaultInstance
    defaultInstance := nil
```

ES6: Class Definition + Instance Methods



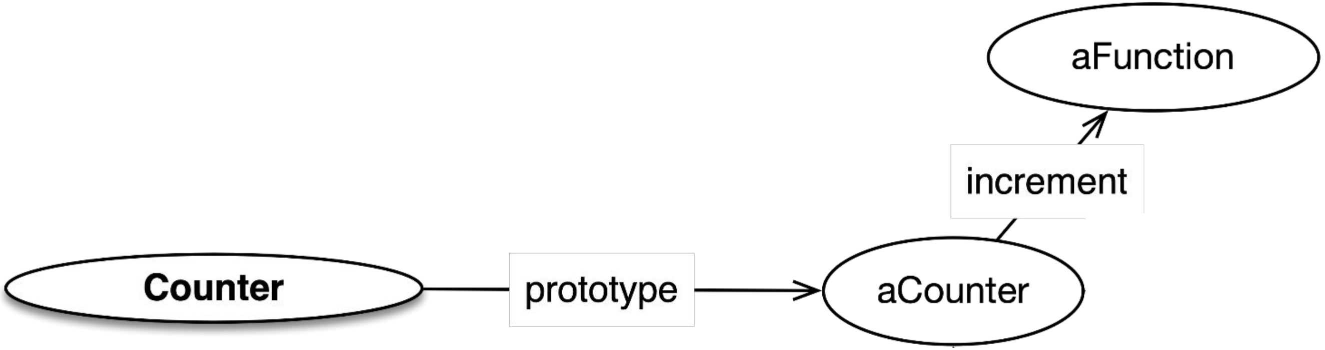
```
class Counter {
    constructor() {
        this.count = 0;
    }
    // Instance methods
    increment() {
        this.count = this.count + 1;
    }
```

ES6: Class Methods

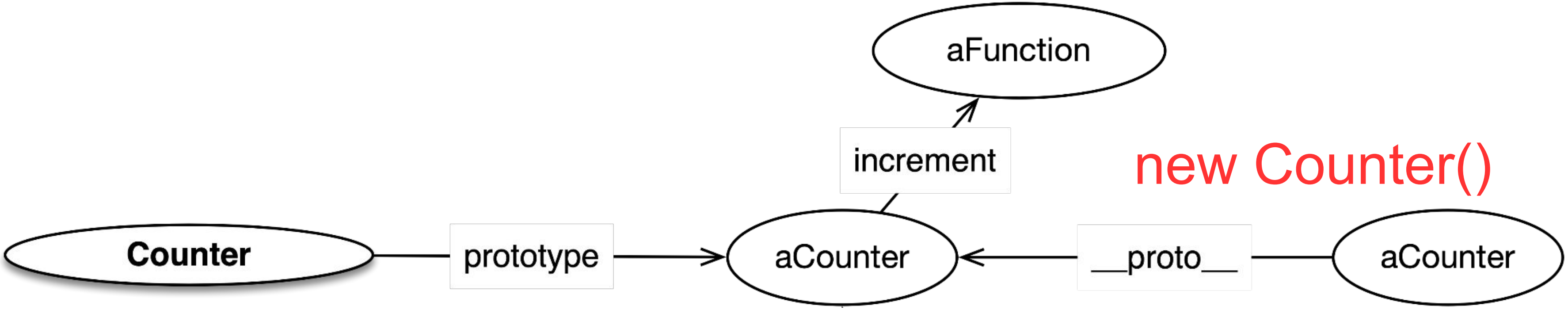


```
static createDefaultInstance(){
    return this.defaultInstance = new this();
}
static getDefaultInstance (){
    if(this.defaultInstance == null){
        return this.createDefaultInstance();
    }
    return this.defaultInstance;
}
static resetDefaultInstance (){
    this.defaultInstance = null;
}
```

JS “Instance Side” Object Graph

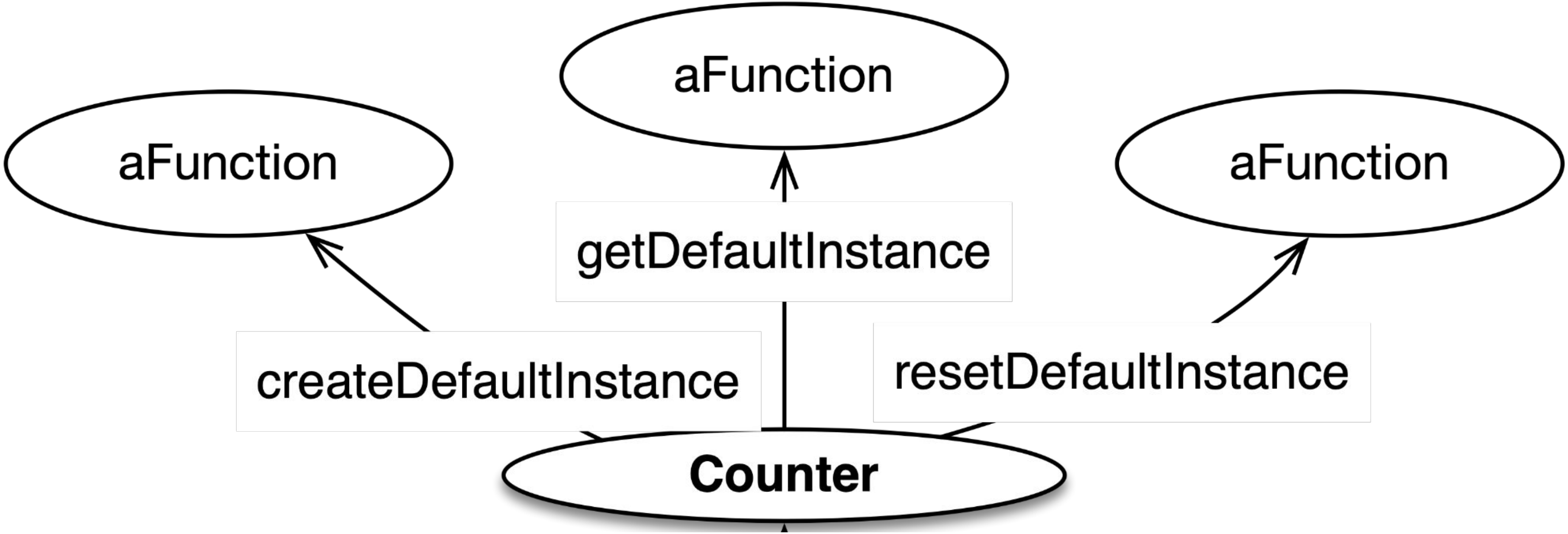


JS “Instance Side” Object Graph

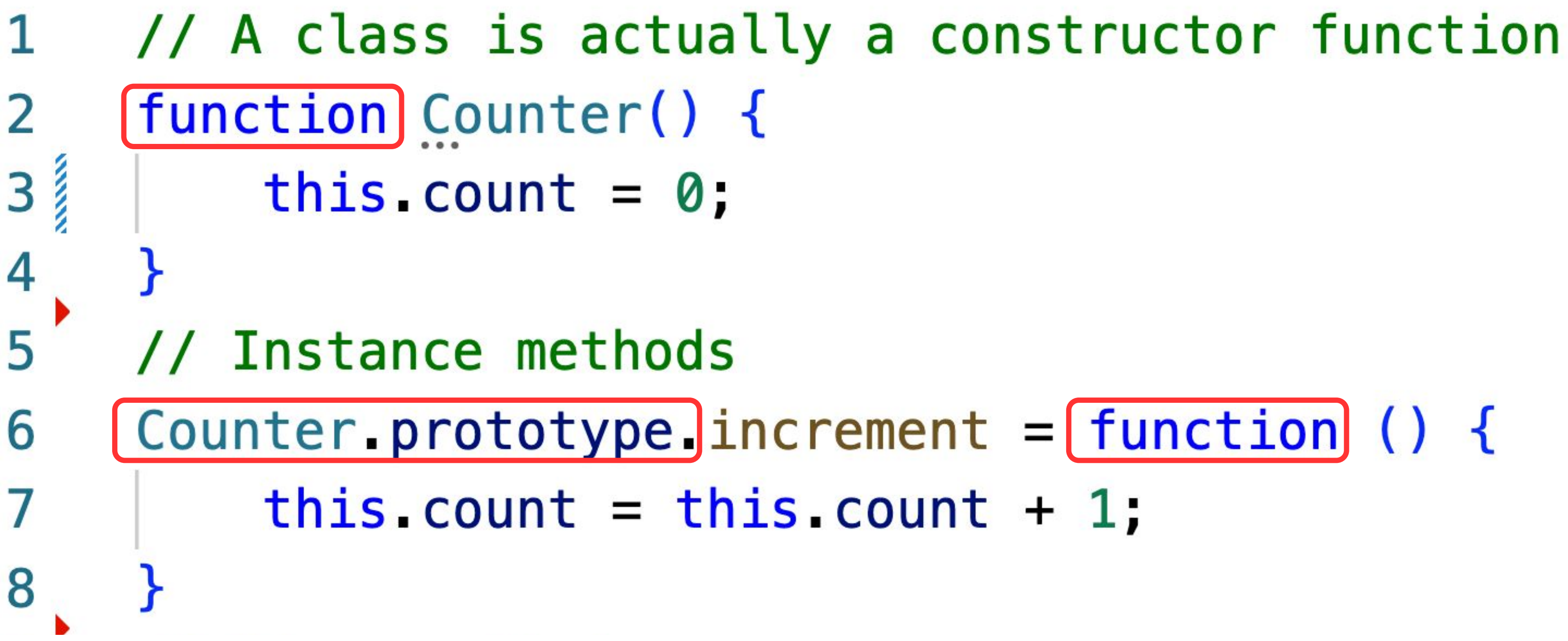


new Counter()

JS “Class Side” Object Graph

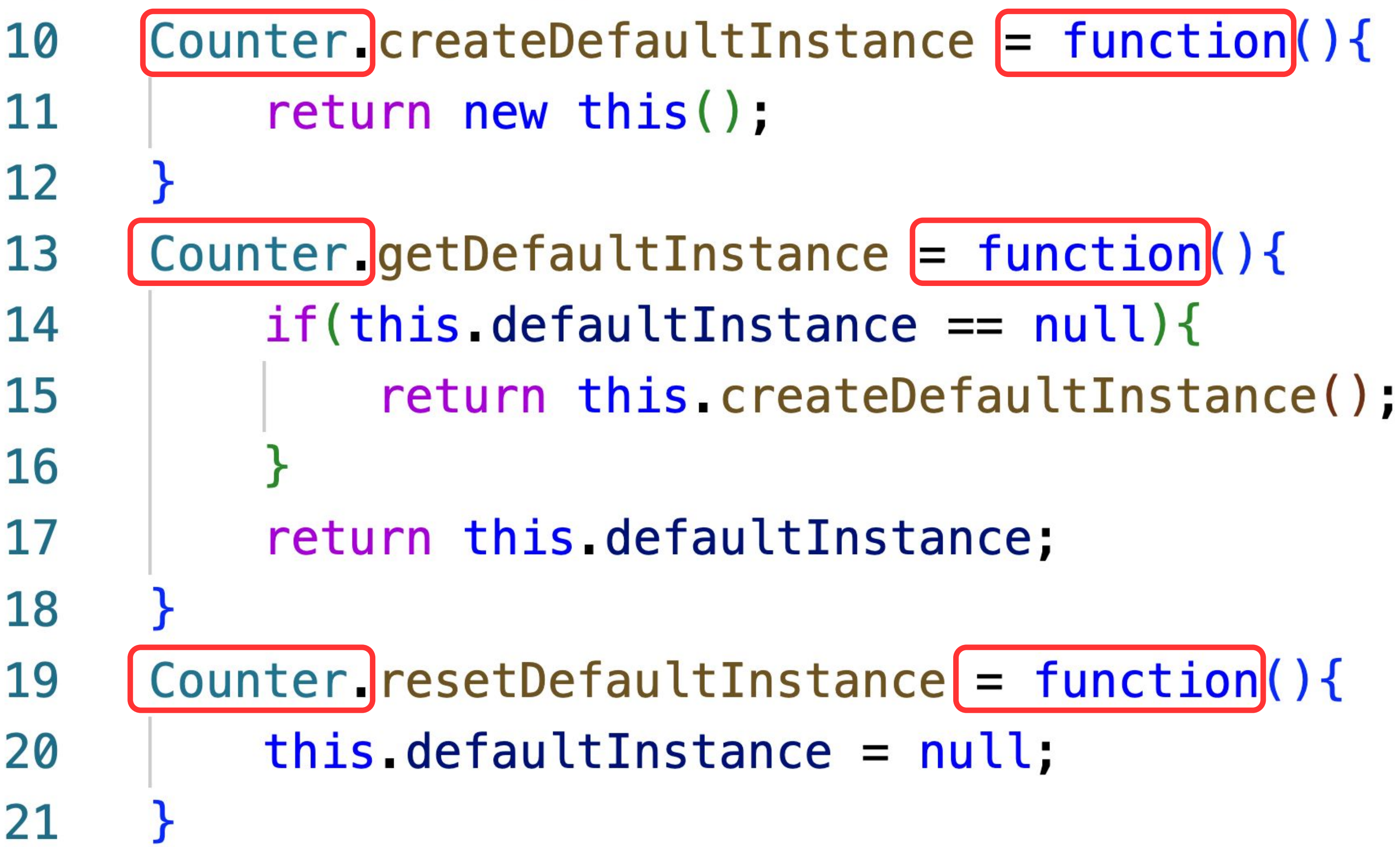


ES5: Class Definition + Instance Methods



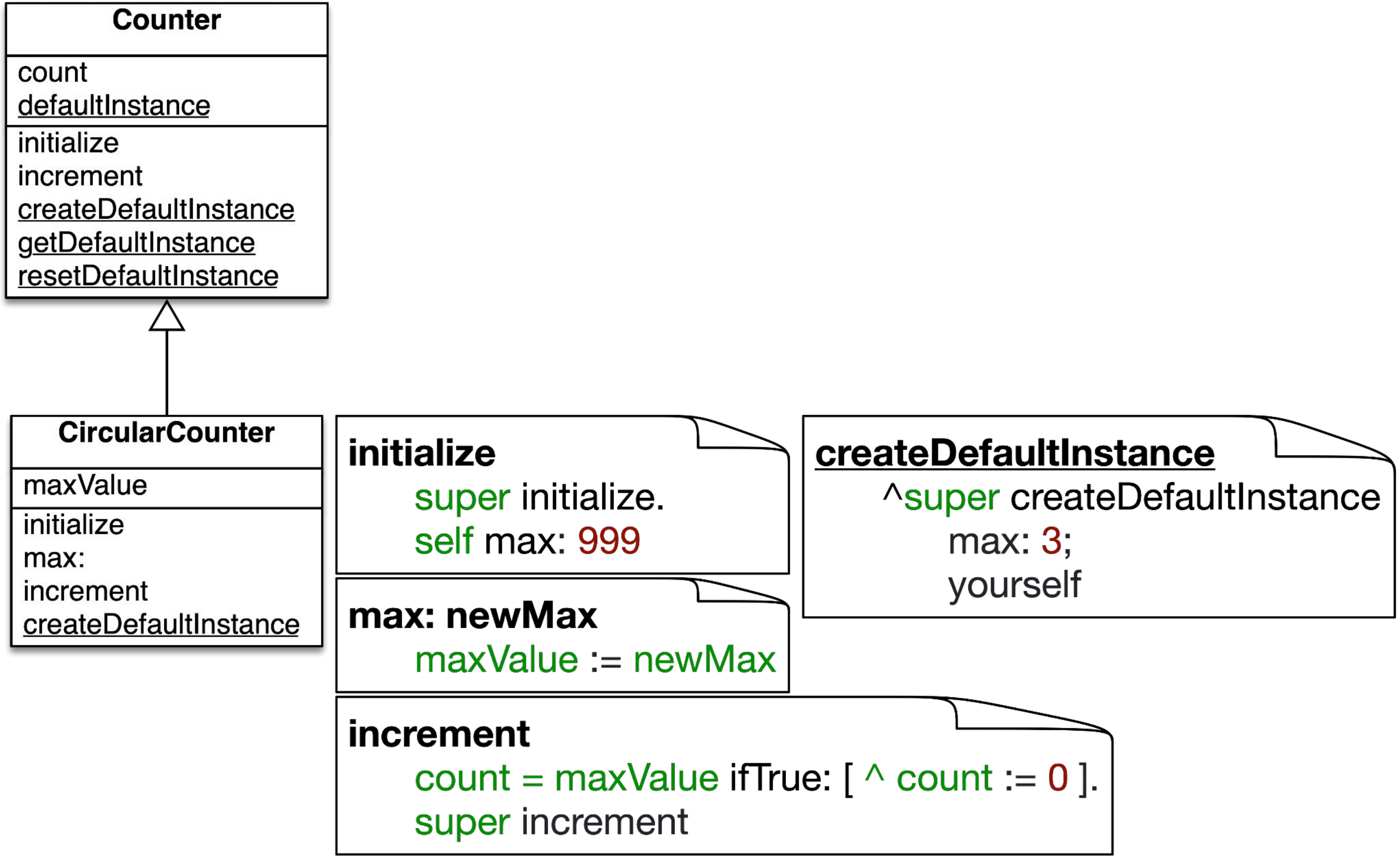
```
// A class is actually a constructor function
function Counter() {
    this.count = 0;
}
// Instance methods
Counter.prototype.increment = function () {
    this.count = this.count + 1;
}
```

ES5: Class Methods



```
Counter.createDefaultInstance = function(){
    return new this();
}
Counter.getDefaultInstance = function(){
    if(this.defaultInstance == null){
        return this.createDefaultInstance();
    }
    return this.defaultInstance;
}
Counter.resetDefaultInstance = function(){
    this.defaultInstance = null;
}
```

Subclass Transpilation by Example



Counter

count
defaultInstance

initialize
increment
createDefaultInstance
getDefaultInstance
resetDefaultInstance

CircularCounter

maxValue

initialize
max:
increment
createDefaultInstance

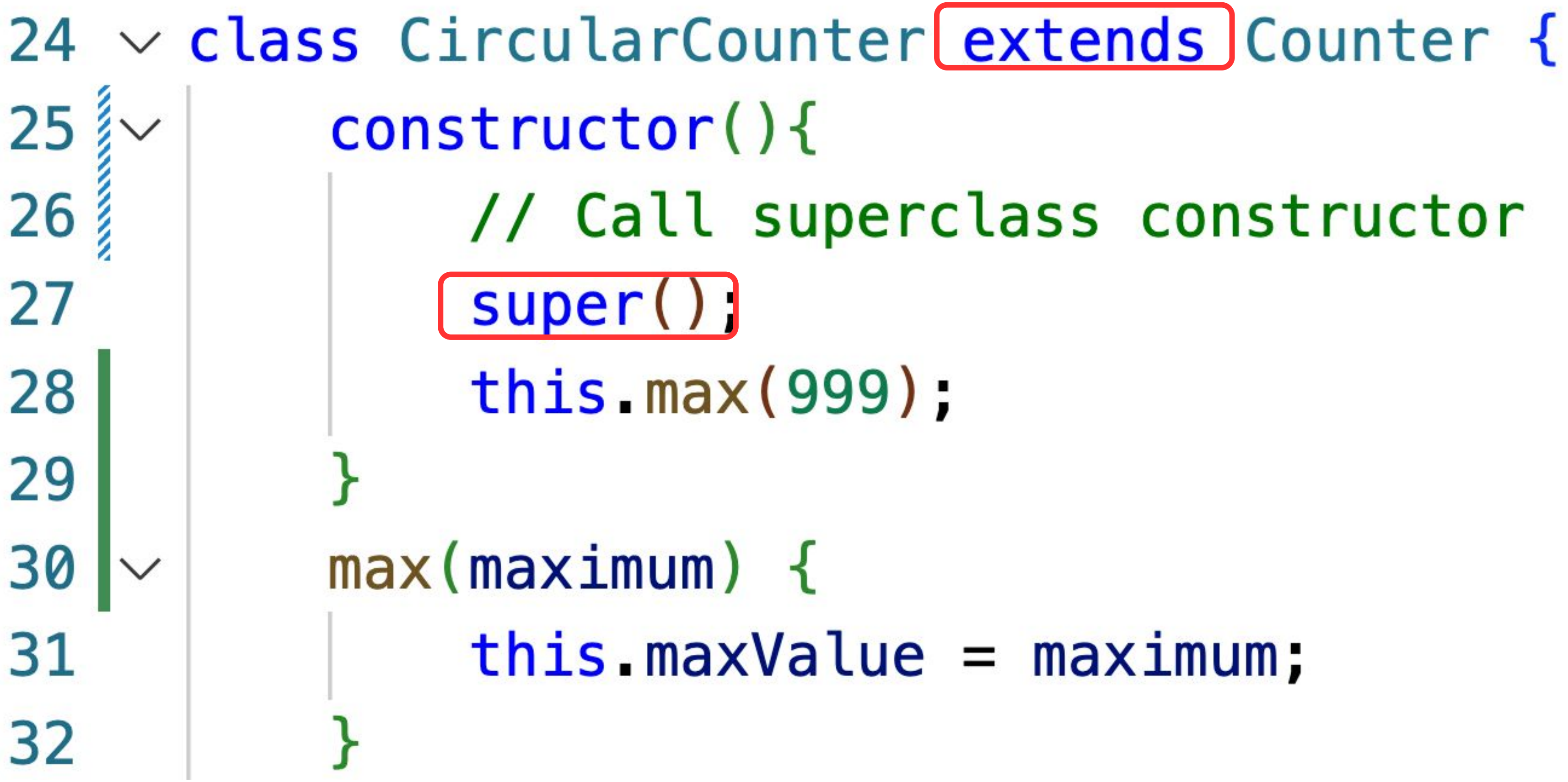
```
initialize
    super initialize.
    self max: 999
```

```
max: newMax
    maxValue := newMax
```

```
increment
    count = maxValue ifTrue: [ ^ count := 0 ].
    super increment
```

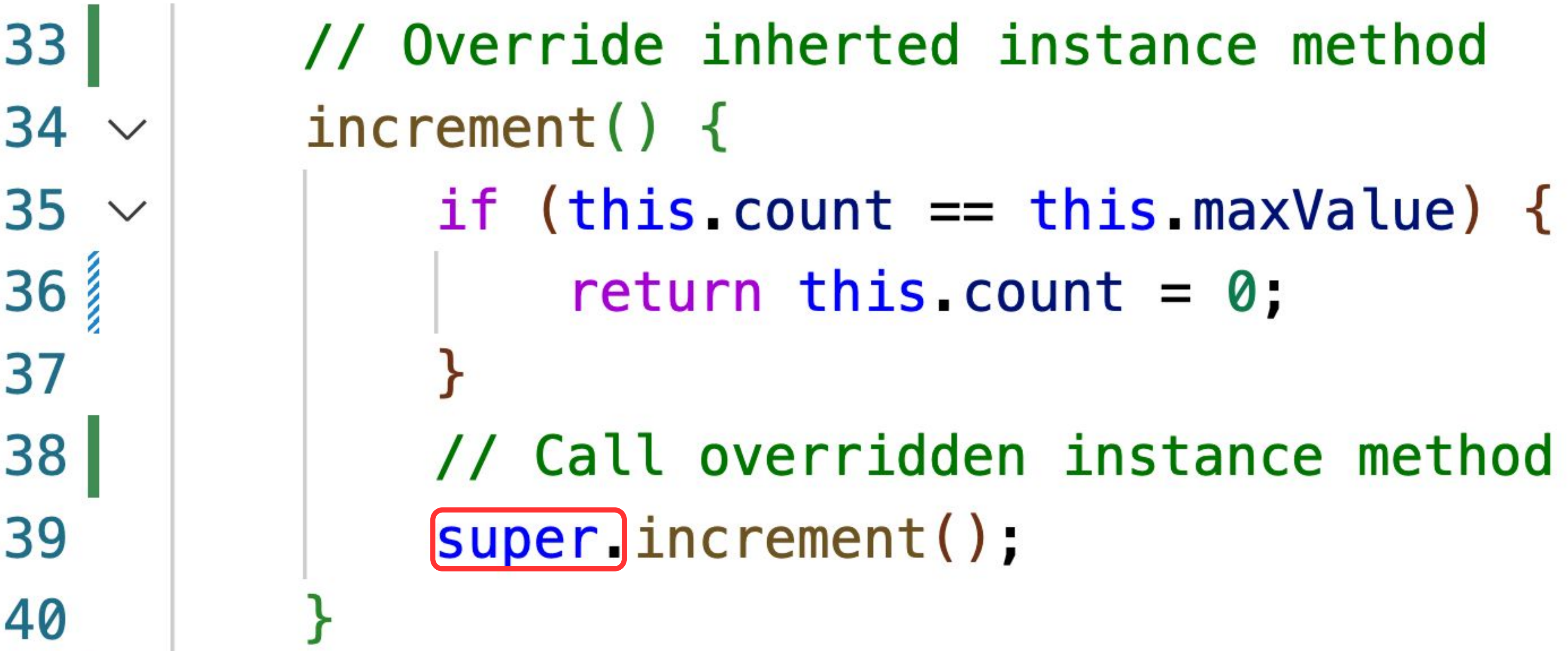
```
createDefaultInstance
    ^super createDefaultInstance
        max: 3;
        yourself
```

ES6: Subclass Definition



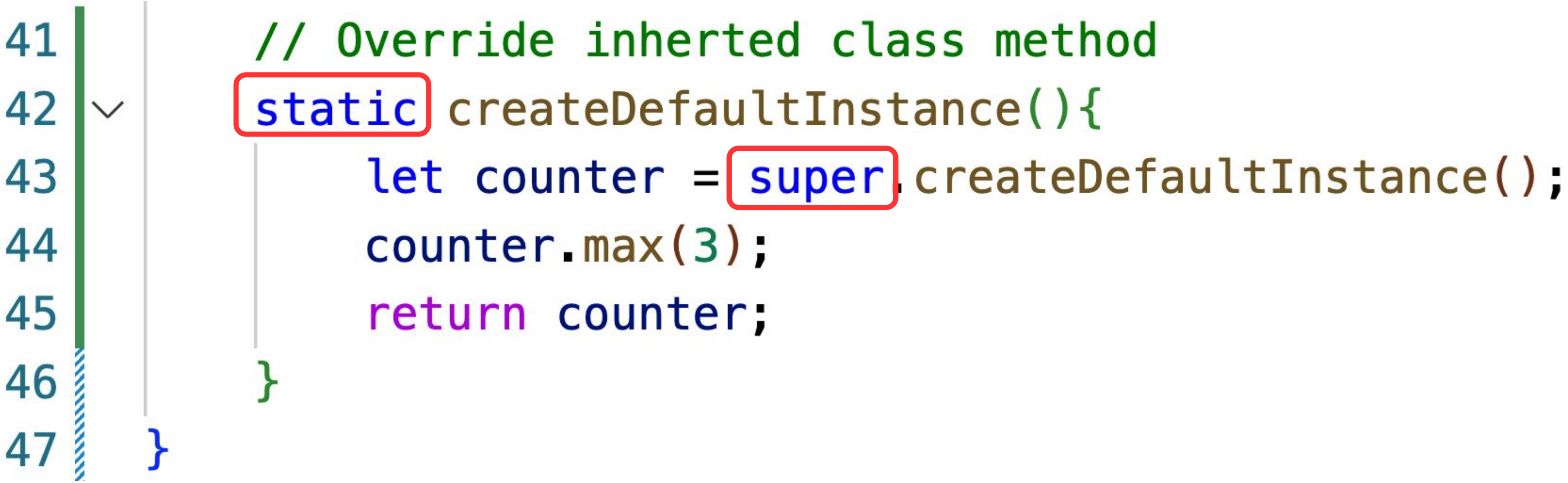
```
class CircularCounter extends Counter {
    constructor(){
        // Call superclass constructor
        super();
        this.max(999);
    }
    max(maximum) {
        this.maxValue = maximum;
    }
```

ES6: Subclass Overriding Instance Methods



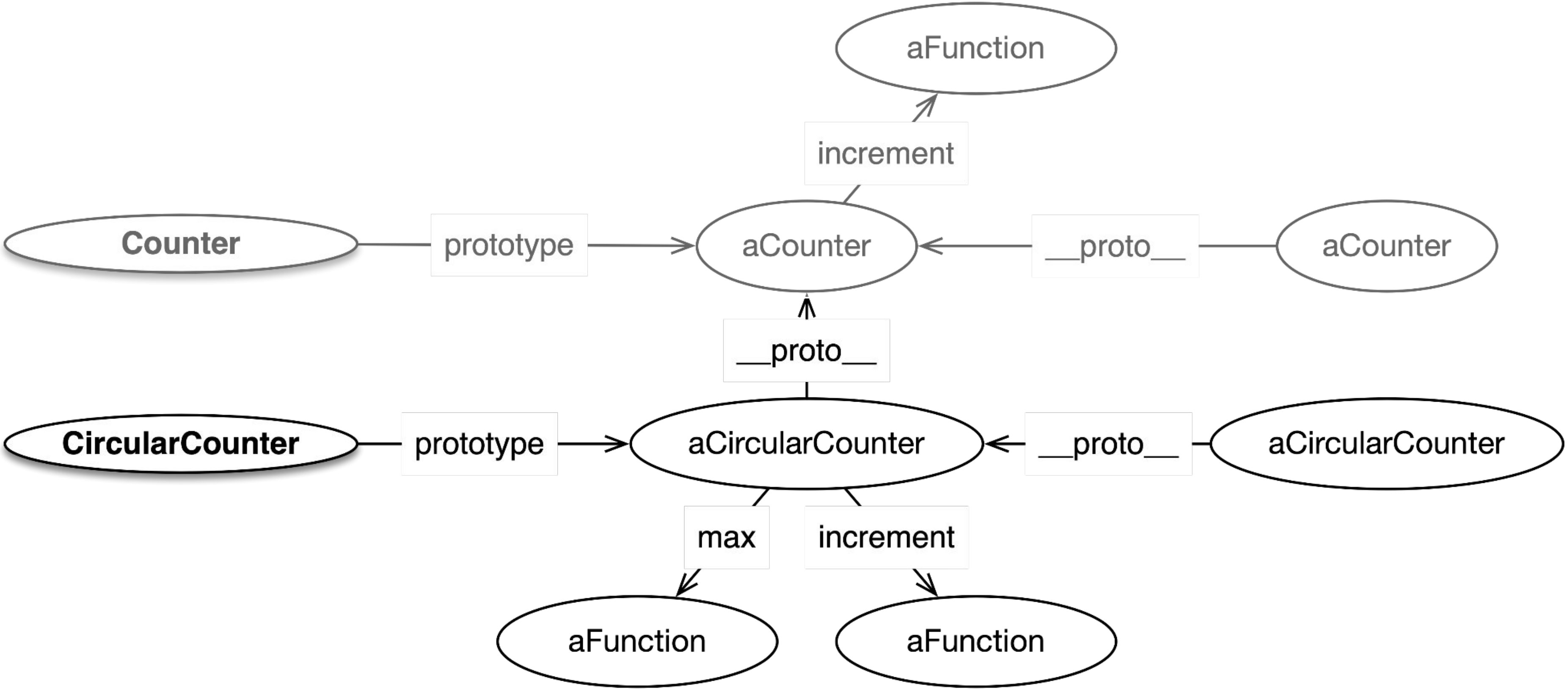
```
// Override inherted instance method
increment() {
    if (this.count == this.maxValue) {
        return this.count = 0;
    }
    // Call overridden instance method
    super.increment();
}
```

ES6: Subclass Methods

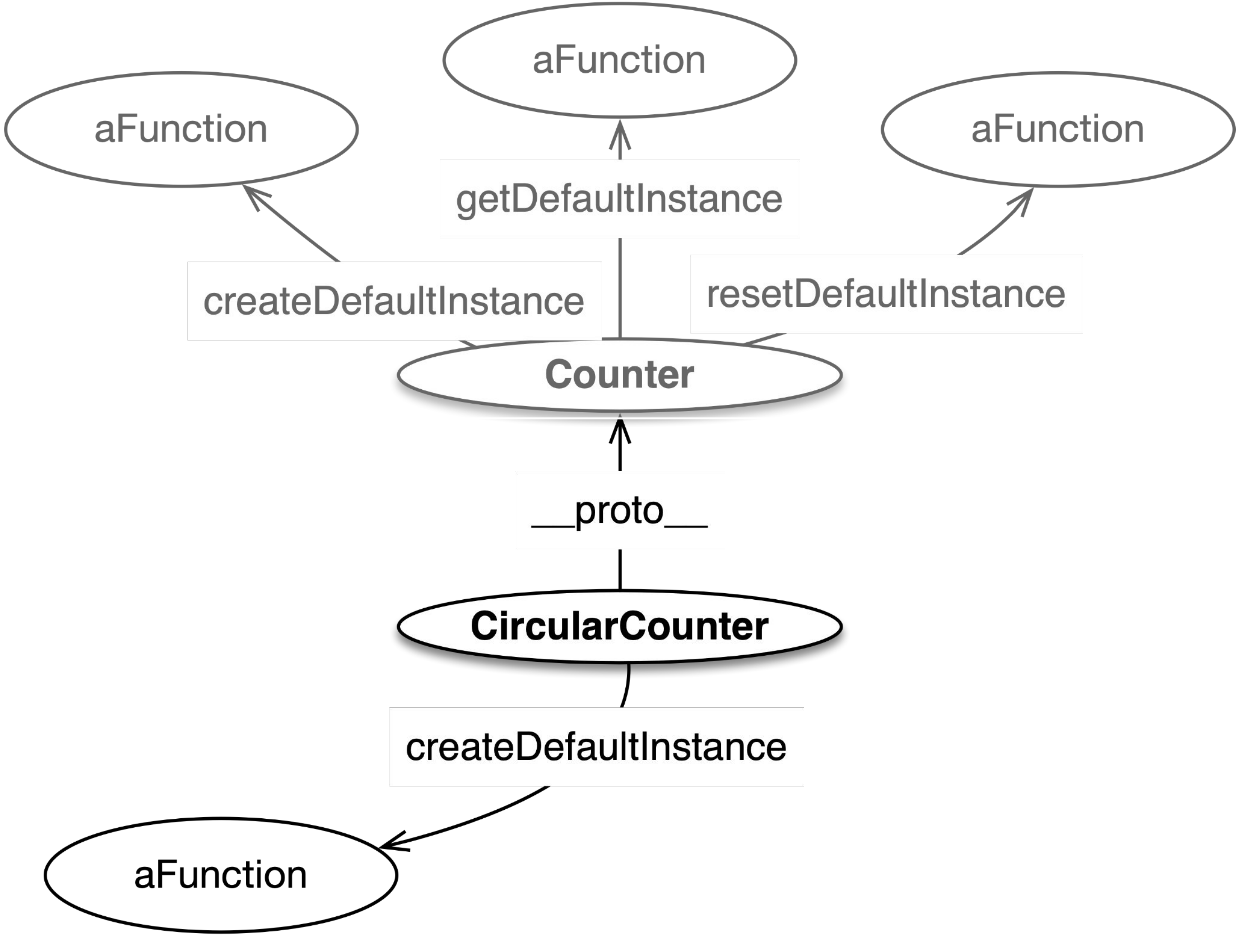


```
    // Override inherted class method
    static createDefaultInstance(){
        let counter = super.createDefaultInstance();
        counter.max(3);
        return counter;
    }
}
```

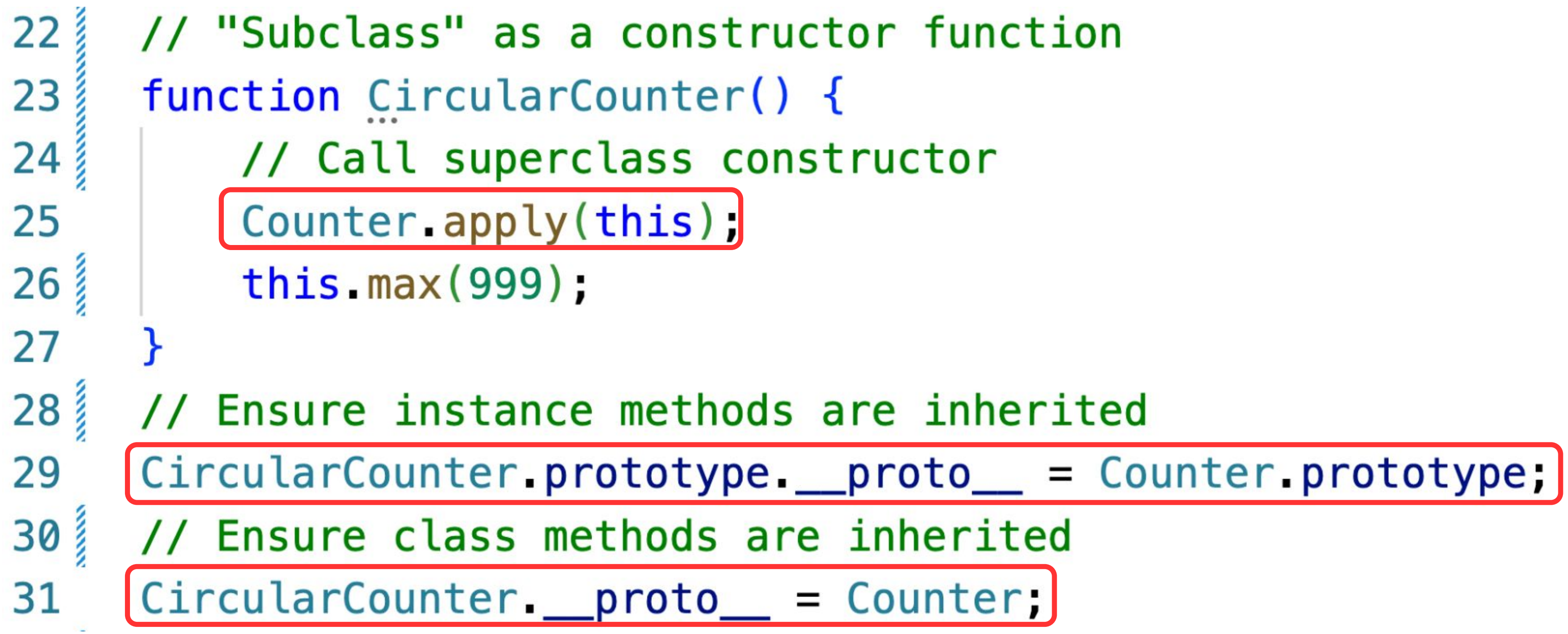
JS Subclass “Instance Side” Object Graph



JS Subclass “Class Side” Object Graph

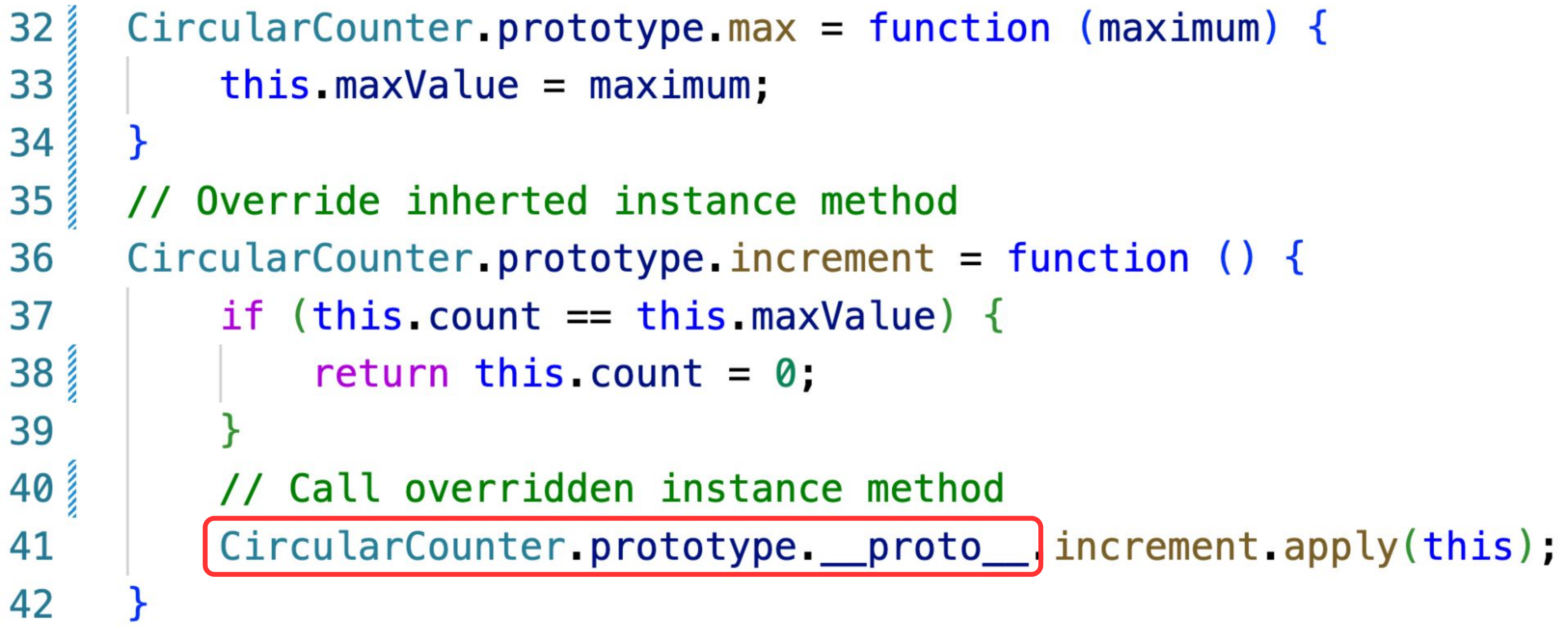


ES5: Subclass Definition



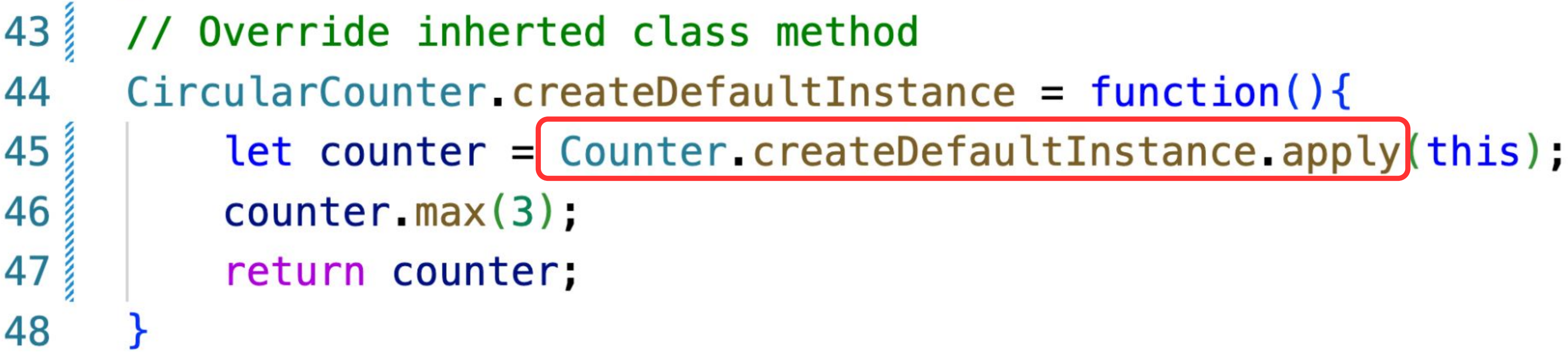
```
// "Subclass" as a constructor function
function CircularCounter() {
    // Call superclass constructor
    Counter.apply(this);
    this.max(999);
}
// Ensure instance methods are inherited
CircularCounter.prototype.__proto__ = Counter.prototype;
// Ensure class methods are inherited
CircularCounter.__proto__ = Counter;
```

ES5: Subclass Instance Methods



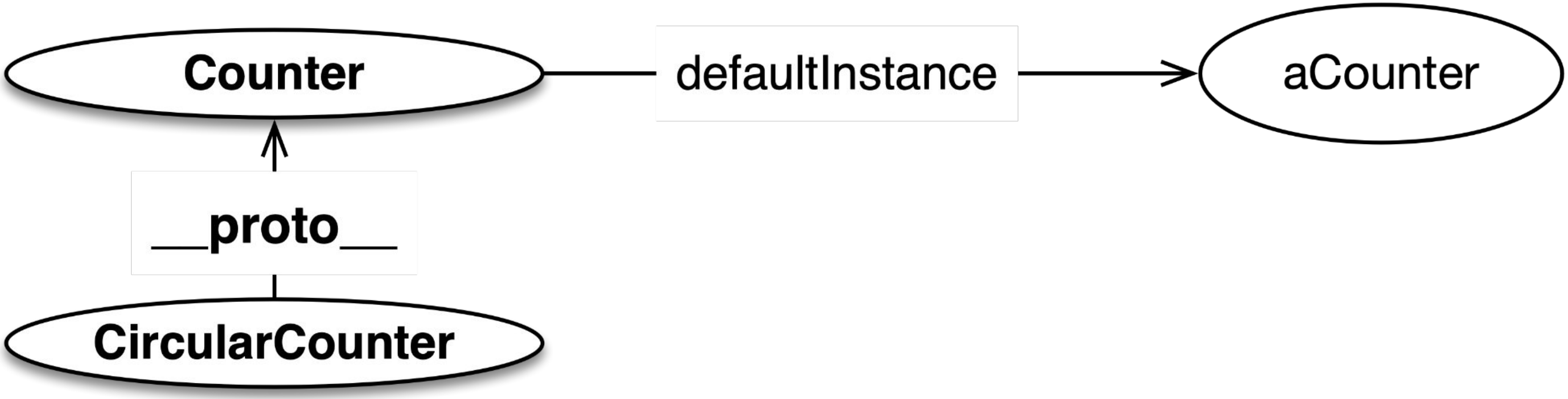
```
CircularCounter.prototype.max = function (maximum) {
    this.maxValue = maximum;
}
// Override inherted instance method
CircularCounter.prototype.increment = function () {
    if (this.count == this.maxValue) {
        return this.count = 0;
    }
    // Call overridden instance method
    CircularCounter.prototype.__proto__.increment.apply(this);
}
```

ES5: Subclass Methods



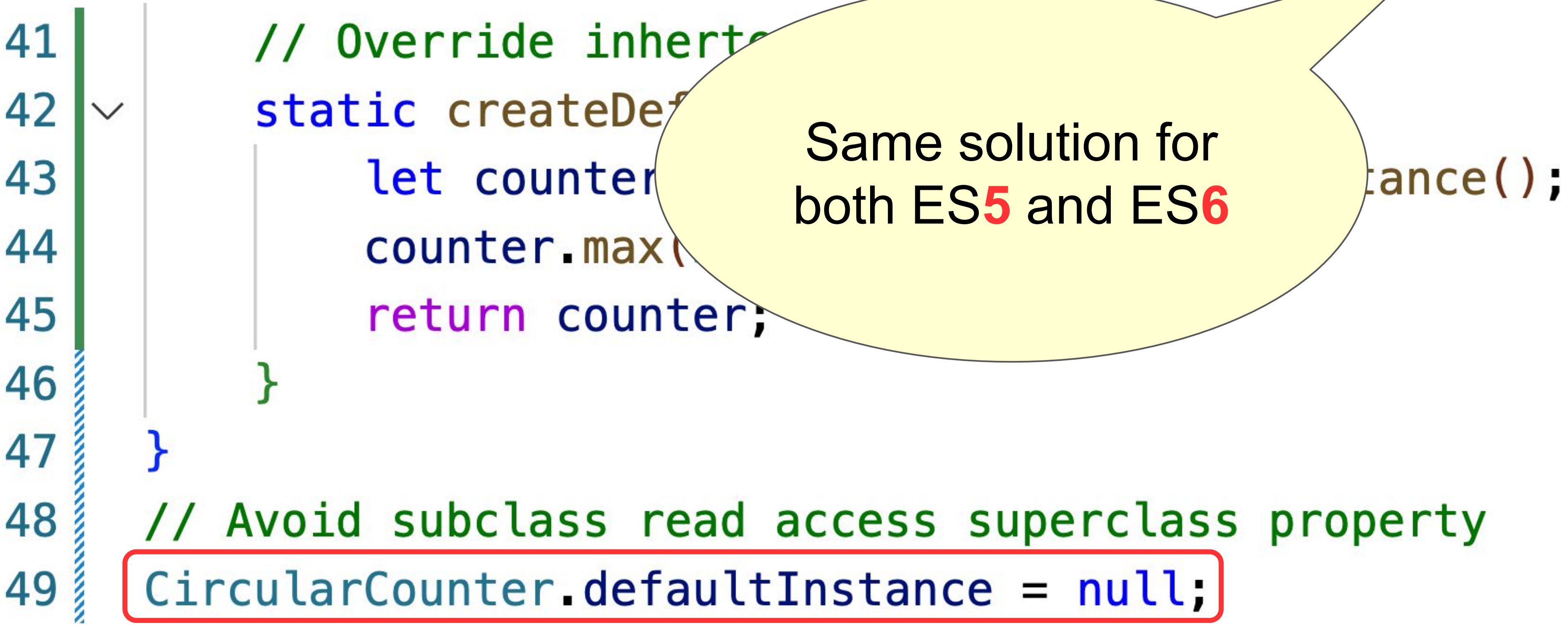
```
// Override inherted class method
CircularCounter.createDefaultInstance = function(){
    let counter = Counter.createDefaultInstance.apply(this);
    counter.max(3);
    return counter;
}
```

JS Subclass Access Superclass IV



```
let c1 = Counter.getDefaultInstance();
let c2 = CircularCounter.getDefaultInstance();
c1 === c2; // true! ❌
```

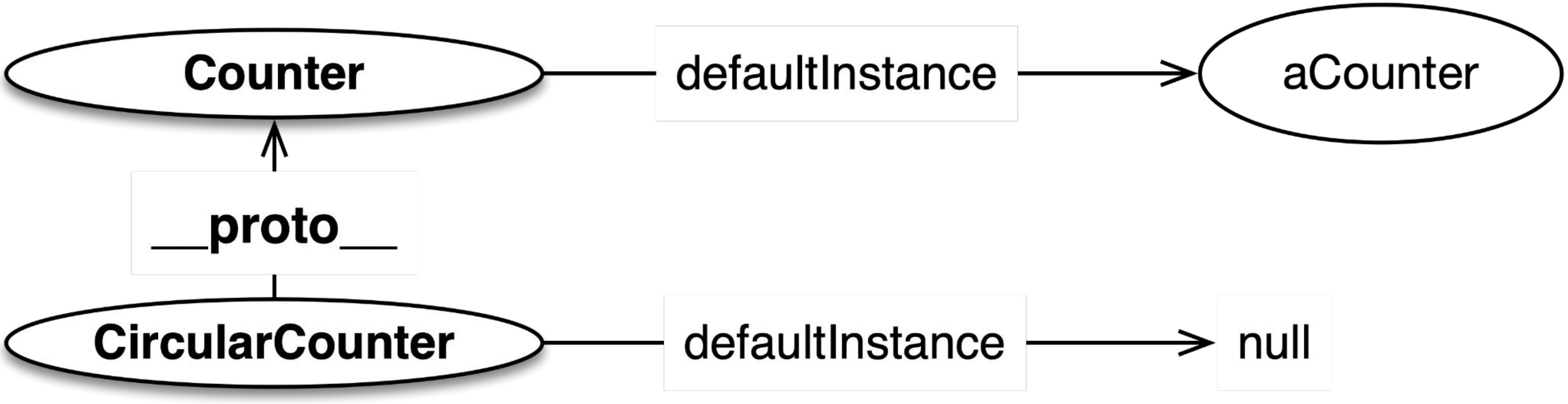
Property Sharing Fix



```
    // Override inherte
    static createDef
        let counter                          ance();
        counter.max(
        return counter;
    }
}
// Avoid subclass read access superclass property
CircularCounter.defaultInstance = null;
```

Same solution for both ES**5** and ES**6**

JS Subclass Access Superclass IV



```
let c1 = Counter.getDefaultInstance();
let c2 = CircularCounter.getDefaultInstance();
c1 === c2; // false! ✓
```

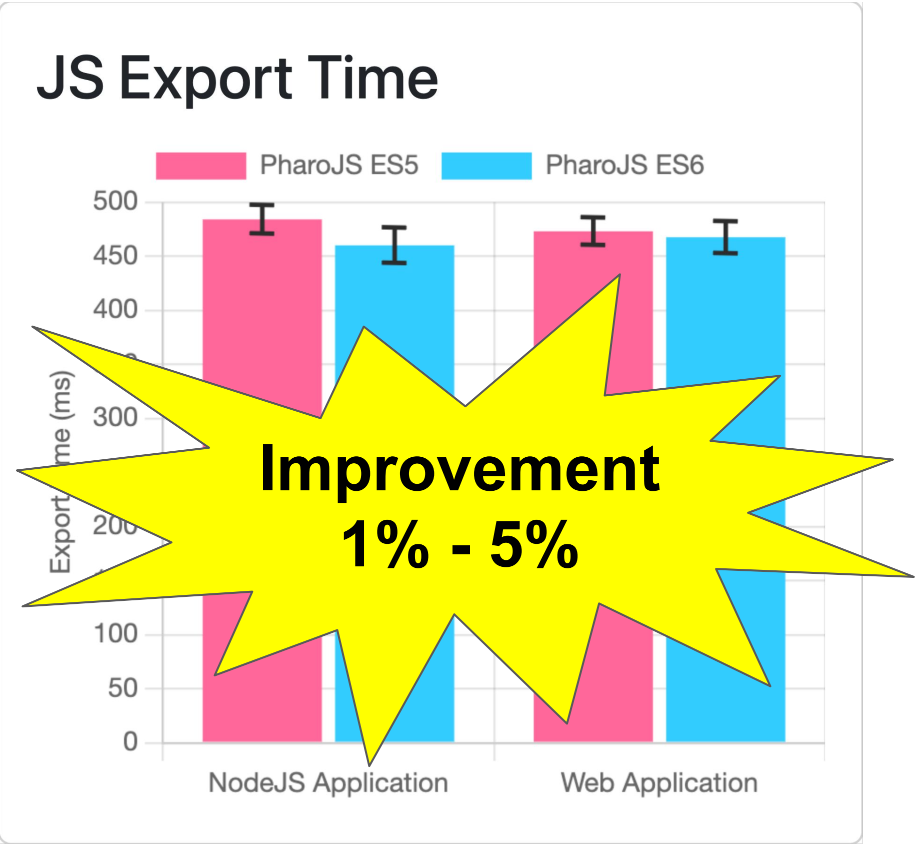
Benchmark Procedure

- Mac Book Pro
  - CPU 8 Intel Core i9, 2.3 GHz,
  - RAM 32 GB, 2667 MHz DDR4
  - Hard drive 1 TB SSD, PCI-Express with APFS File System
  - Mac OS X Ventura 13.2.1

- Pharo 10
- Pharo VM 100 Darwin x86 64-bit

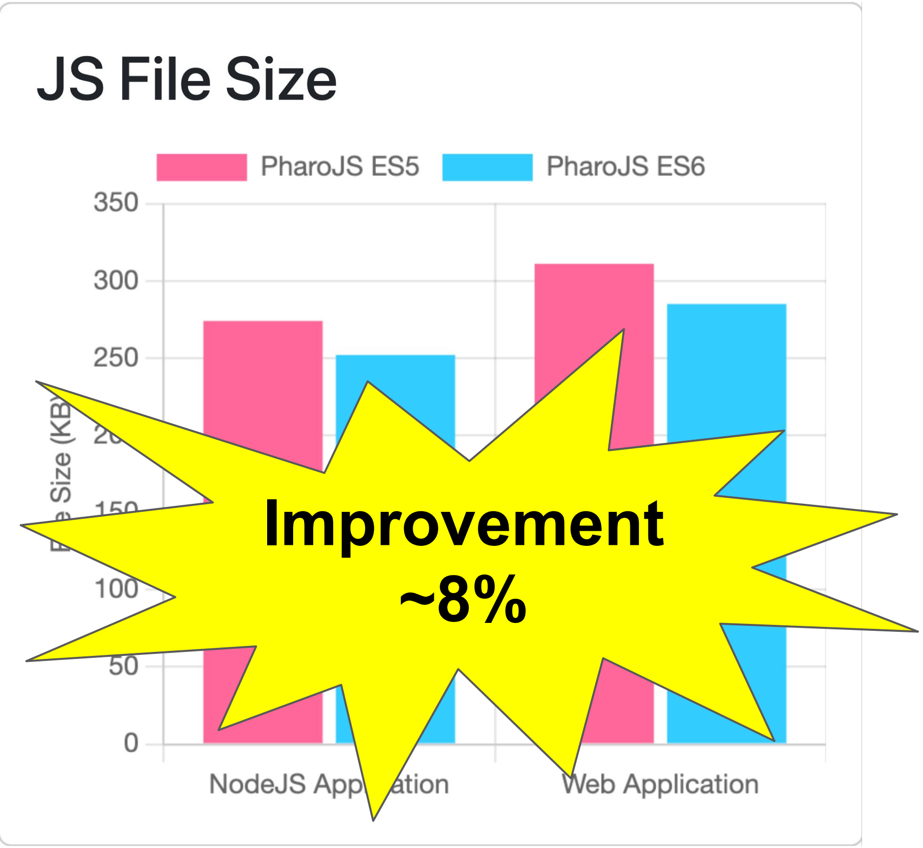
- JS Targets
  - Node
  - Web Browser

Improved Transpilation Time + File Size



JS Export Time

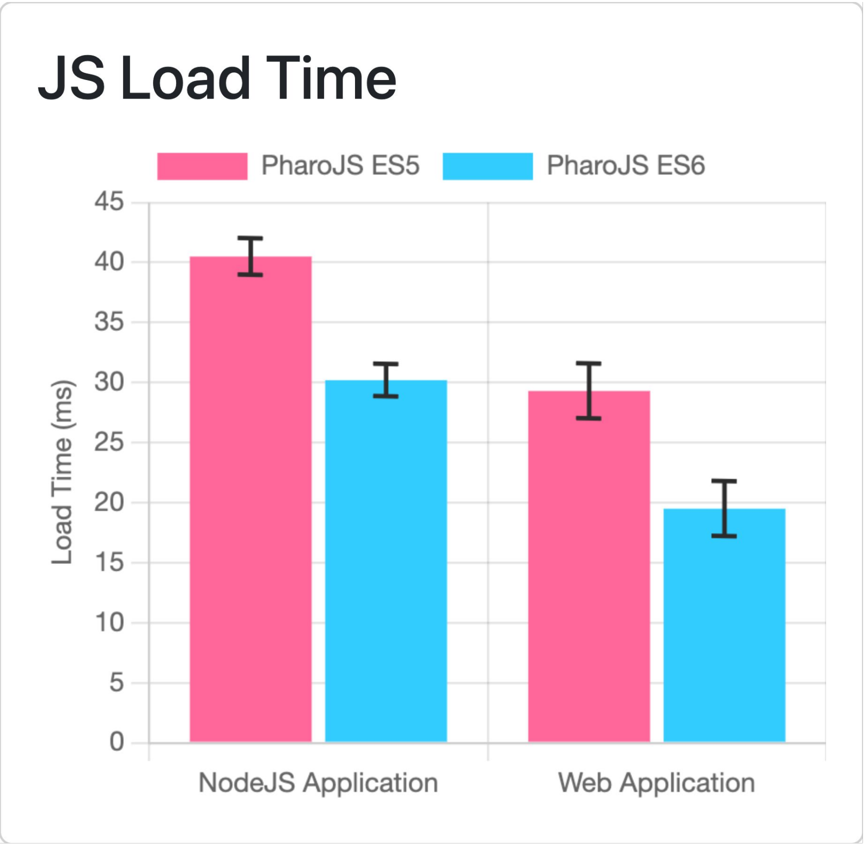
Improvement
1% - 5%



JS File Size

Improvement
~8%

Significantly Faster Load Time

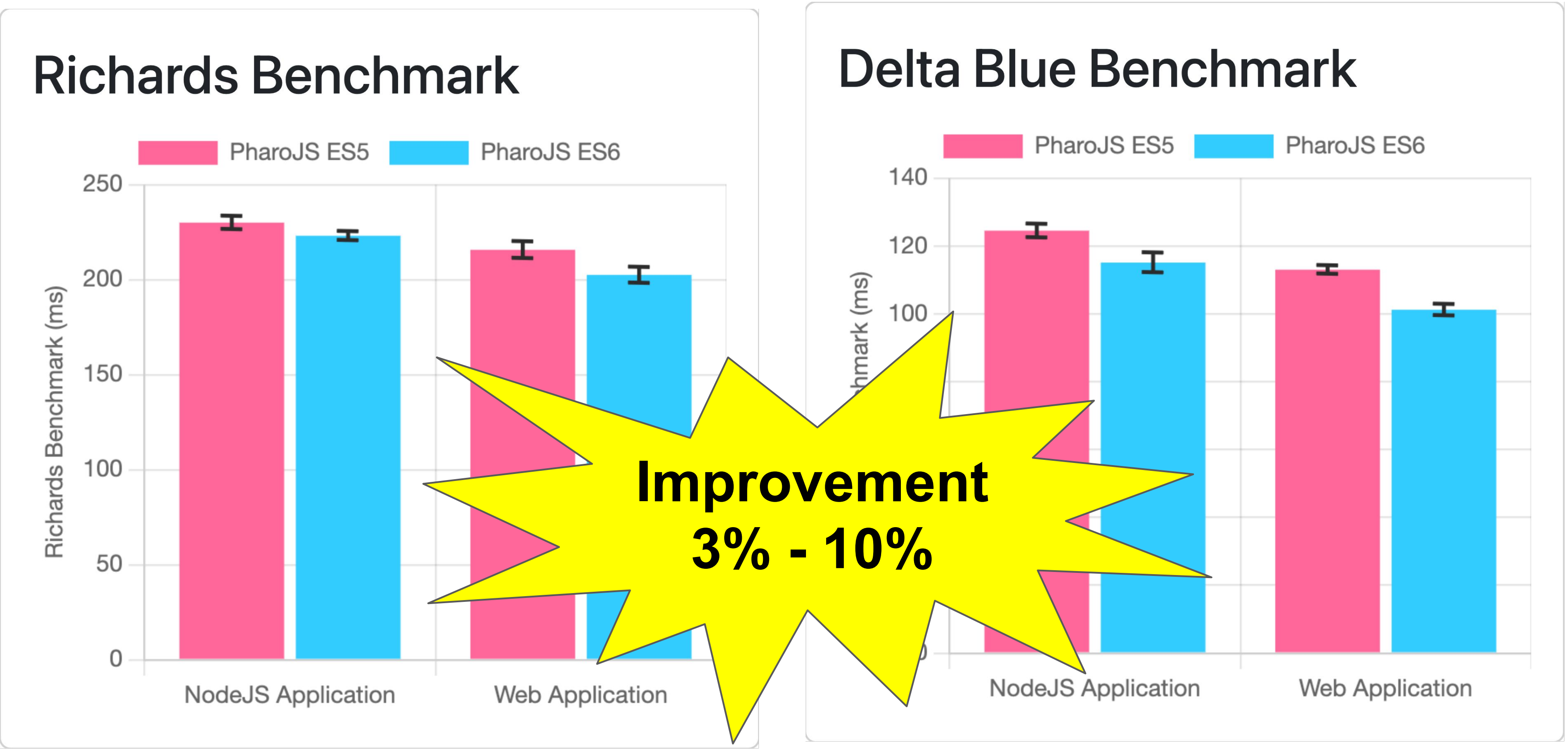


Improvement 25% - 33%

Run-time Benchmark Procedure

- 5 warm up runs
- 10 runs
- Richards: 50 iterations / run
- Delta Blue: 300 iterations / run

Improved Runtime Performance

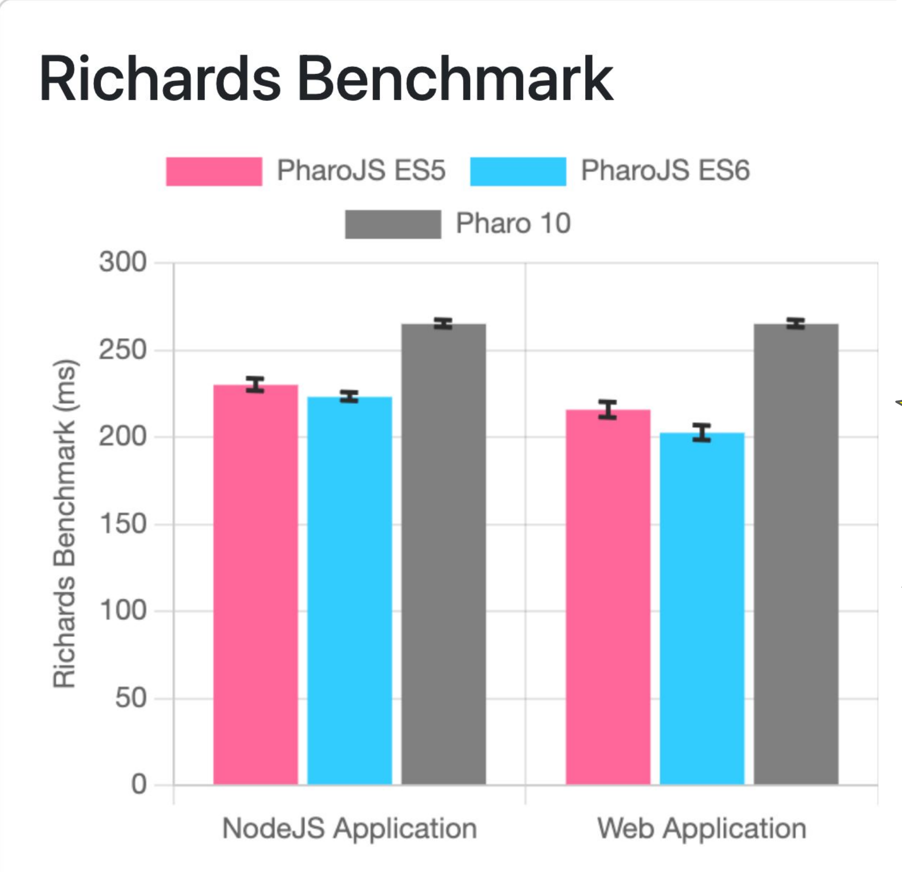


Richards Benchmark

Delta Blue Benchmark

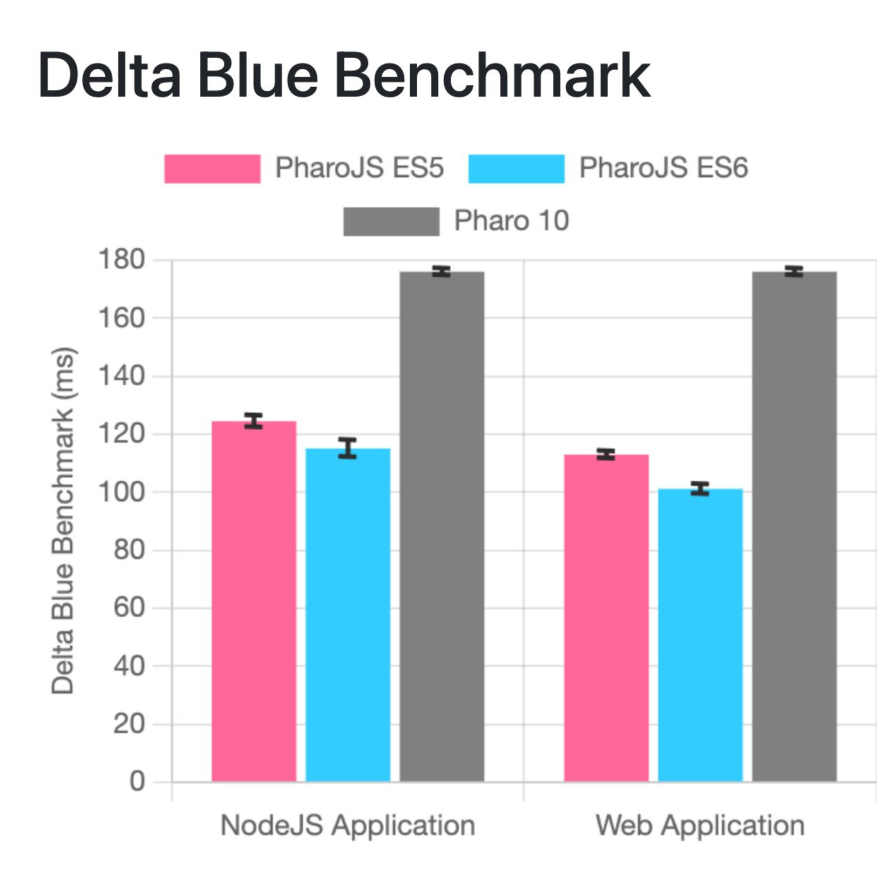
Improvement 3% - 10%

Improved Runtime Performance vs Pharo 10



ES6 vs Pharo10 Improvement 16% - 24%

Improved Runtime Performance vs Pharo 10



ES6 vs Pharo10 Improvement 35% - 43%

Summary

- PharoJS is a viable solution to reuse JS Ecosystem
- Transition from ES5 to ES6 is Beneficial
    - Significantly faster Load time
    - Improved other benchmarks
    - More idiomatic code with ES6

Getting exact Smalltalk Semantics is Still tricky

- ✓ JS white box model = no encapsulation
  - Generate accessors on the fly for third party classes
- ✓ Inherited Instance Variables e.g CircularCounter example
  - Force IV Creation
- ✓ Metaclass inheritance for third party classes
  - **class X {...}** *vs* **class X extends Object {...}**
- Support full Pharo is still a Challenge
  - ✓ DoesNotUnderstand
  - ✓ superclass - subclasses relationship
  - ❓ thisContext, become:, …

Kindly Supported by

EUROPEAN SMALLTALK USER GROUP
esug
www.esug.org

PharoJS.org

Develop in Pharo, Run on JavaScript

GitHub

Thanks to all the contributors!