KINGSLAND UNIVERSITY

Prototypes and Inheritance

Class Inheritance, Prototypes, Prototype Chain

Table of Contents

Have a Question?

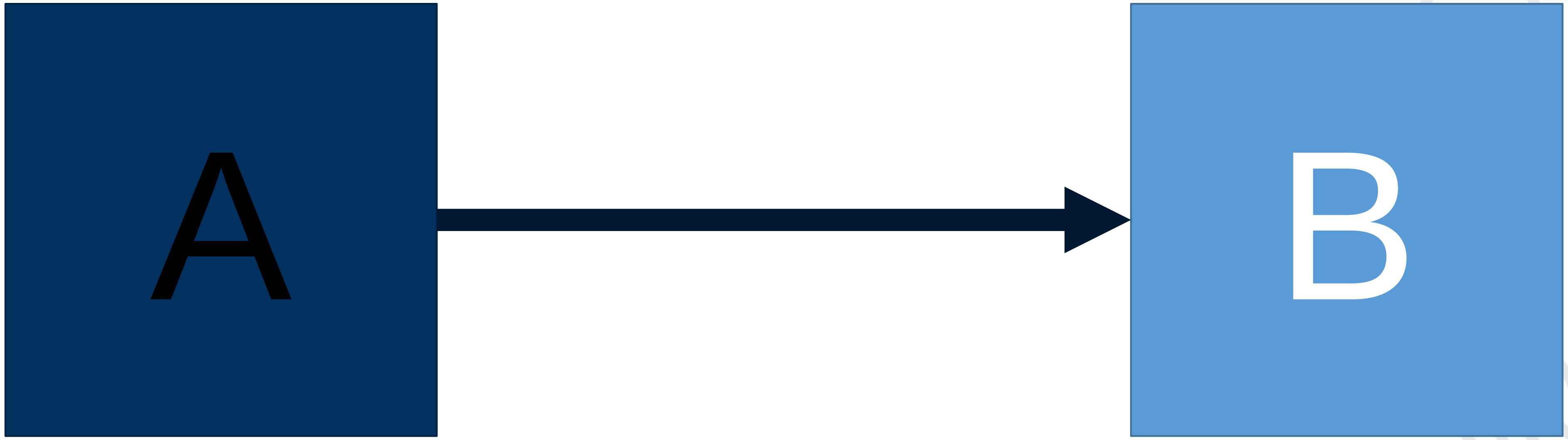
#js-advanced

Inheritance

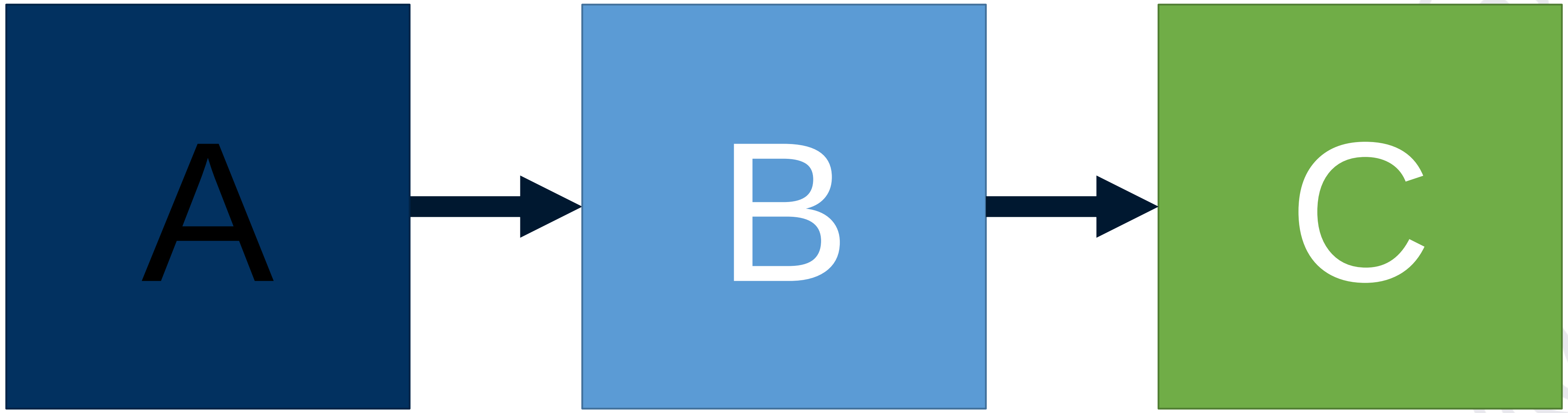
Types of Inheritance

- Simple
- Multilevel
- Hierarchal
- Multiple
- Hybrid

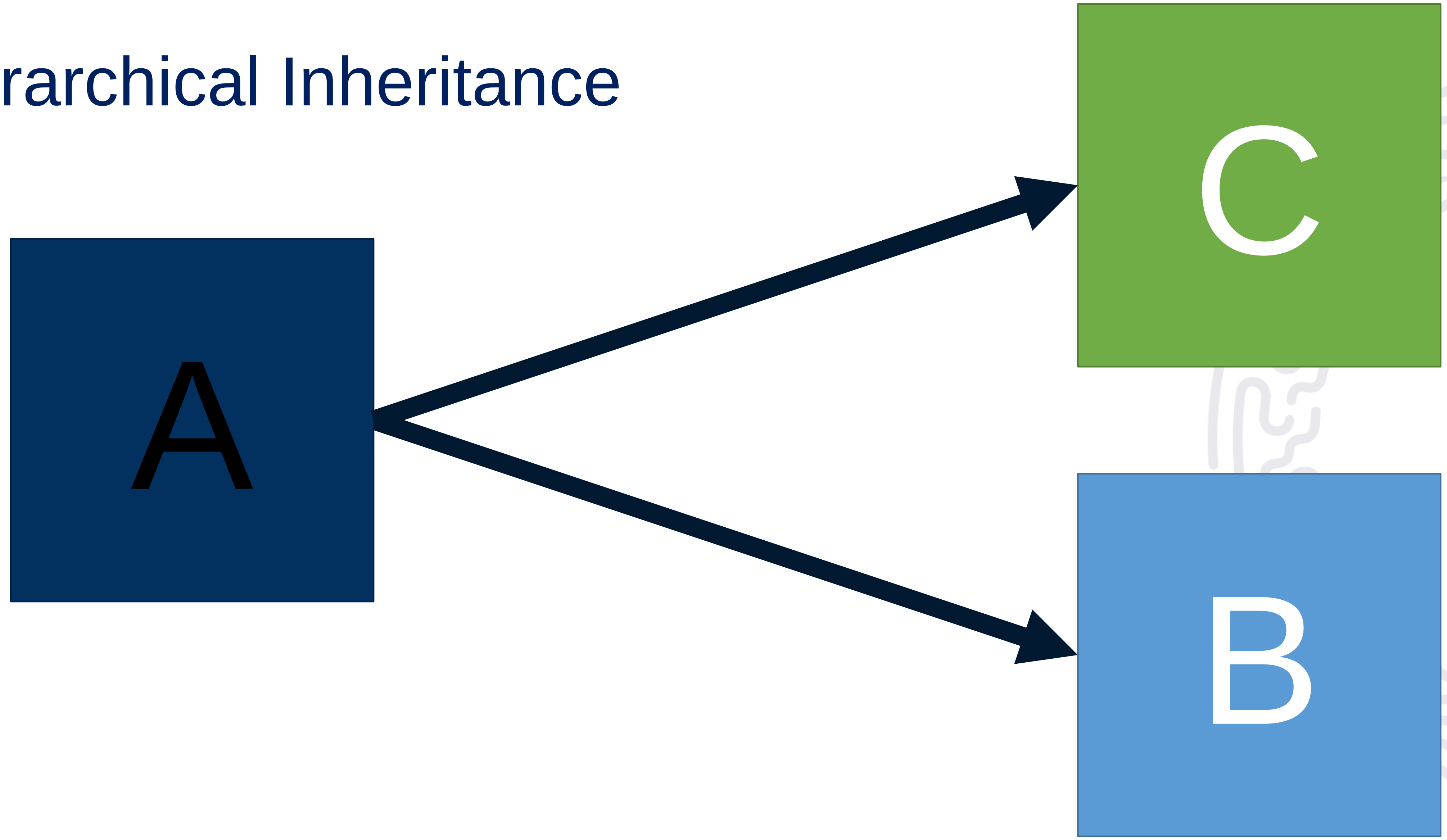
Simple Inheritance



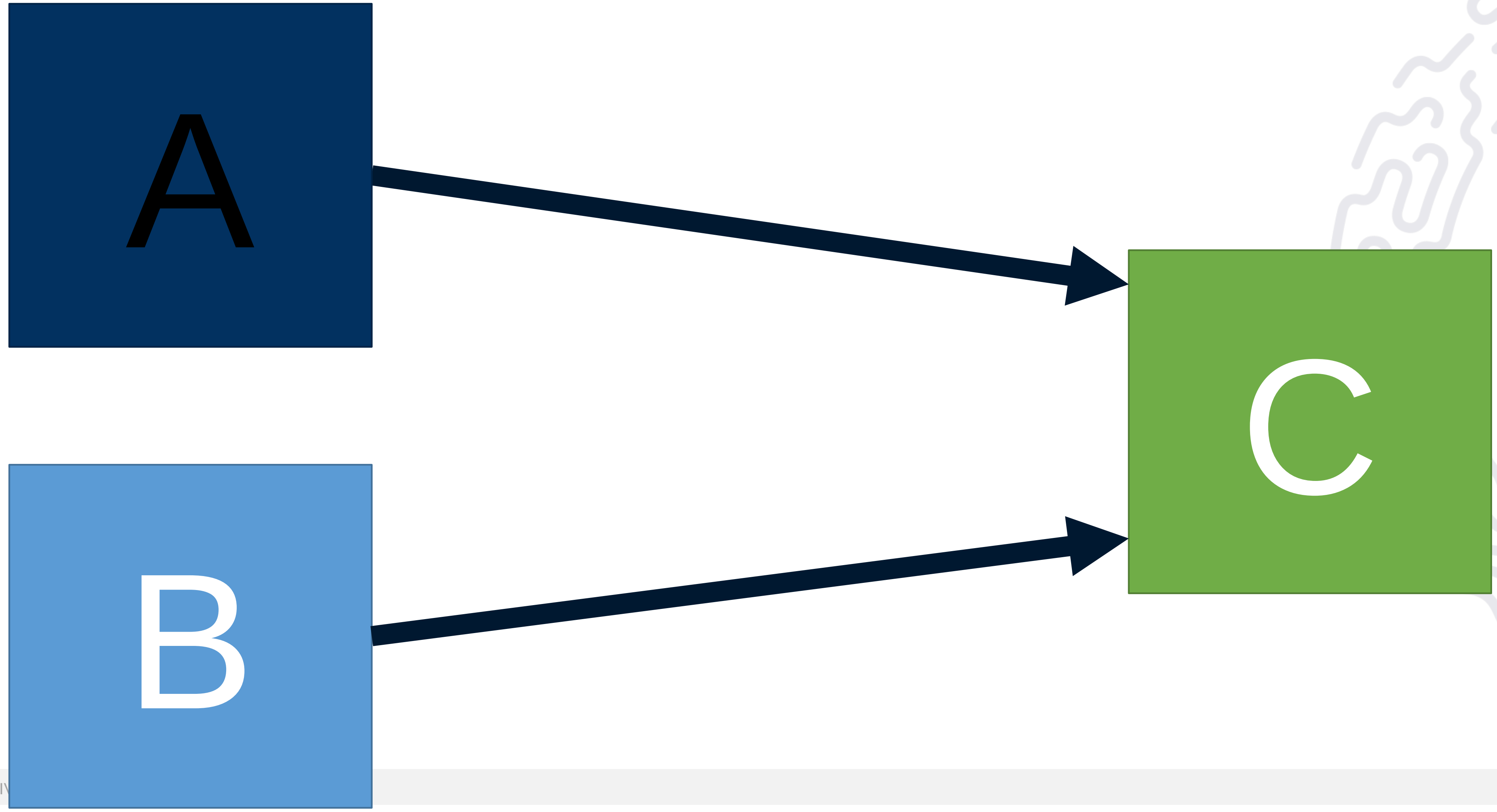
Multilevel Inheritance



Hierarchical Inheritance



Multiple Inheritance

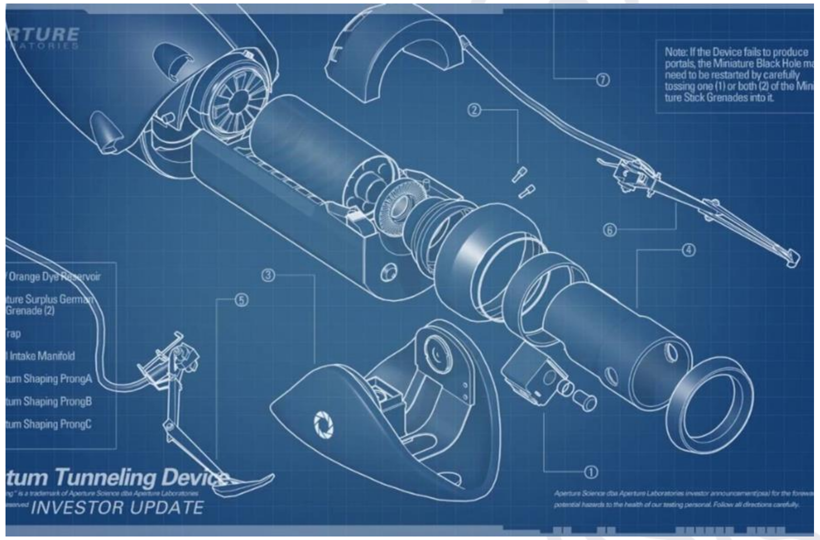


Inheriting Data and Methods

Classical Inheritance

Traditional Classes

- Classes are a **design pattern**
- Classes mean - creating **copies**
  - When **instantiated** - a **copy** from class to instance
  - When **inherited** - a **copy** from parent to child



Classes in JavaScript

- **Prototypal inheritance** instead of classical inheritance
- **Does not automatically** create copies
- Common keys and values are shared by **reference**
- **Delegates not blueprints!**

Class Syntax - Example

```
class Foo {
    constructor(who) {
        this.who = who;
    }
    identify() { return "I am " + this.me; }
}
```

class Bar inherits Foo

```
class Bar extends Foo {
    constructor(who) {
        super(who);
    }
    speak() {
        console.log("Hello, " + this.identify() + ".");
    }
}
```

Invoke the parent constructor

Prototype Inheritance

```
function Foo(who) {
    this.me = who;
}
Foo.prototype.identify = function () { return "I am " + this.me; }
function Bar(who) { Foo.call(this, who); }

Bar.prototype = Object.create(Foo.prototype);
Bar.prototype.speak = function () {
    console.log("Hello, " + this.identify() + ".");
}
let b1 = new Bar("b1");
let b2 = new Bar("b2");
b1.speak(); b2.speak();
```

How Does It Work?

The Prototype Chain

JavaScript Objects

- Literals

```
let bar = {
  me: "I am b1",
  speak: function() {
   console.log("Hello, " +
     this.me + ".");
  }
};
```

- Constructed

```
function Bar(name) {
  this.me = "I am " + name;
  this.speak = function() {
   console.log("Hello, " +
     this.me + ".");
  };
};  let b1 = new Bar("b1");
```

What is a Prototype?

- Just an **object**
- **Internal property**
  - Used to implement **prototype- based inheritance** and shared properties
- **Reference** to another objects
  - Objects are **not** separate and disconnected, but **linked**

Object Creation

- **Literal** creation
- **Constructor** creation
  - Have an **implicit reference** (prototype) to the value of their constructor's "prototype" property
  - Gets an internal **__proto__ link** to the object

__proto__ vs Prototype Property

- **`__ proto__`**
  - Property of an objects that **points** at the prototype that has been **set**
  - Using **`__proto__`** directly is deprecated!
- prototype
  - Property of **a function** set if your object is created by a **constructor function**
  - Objects do not have **prototype** property

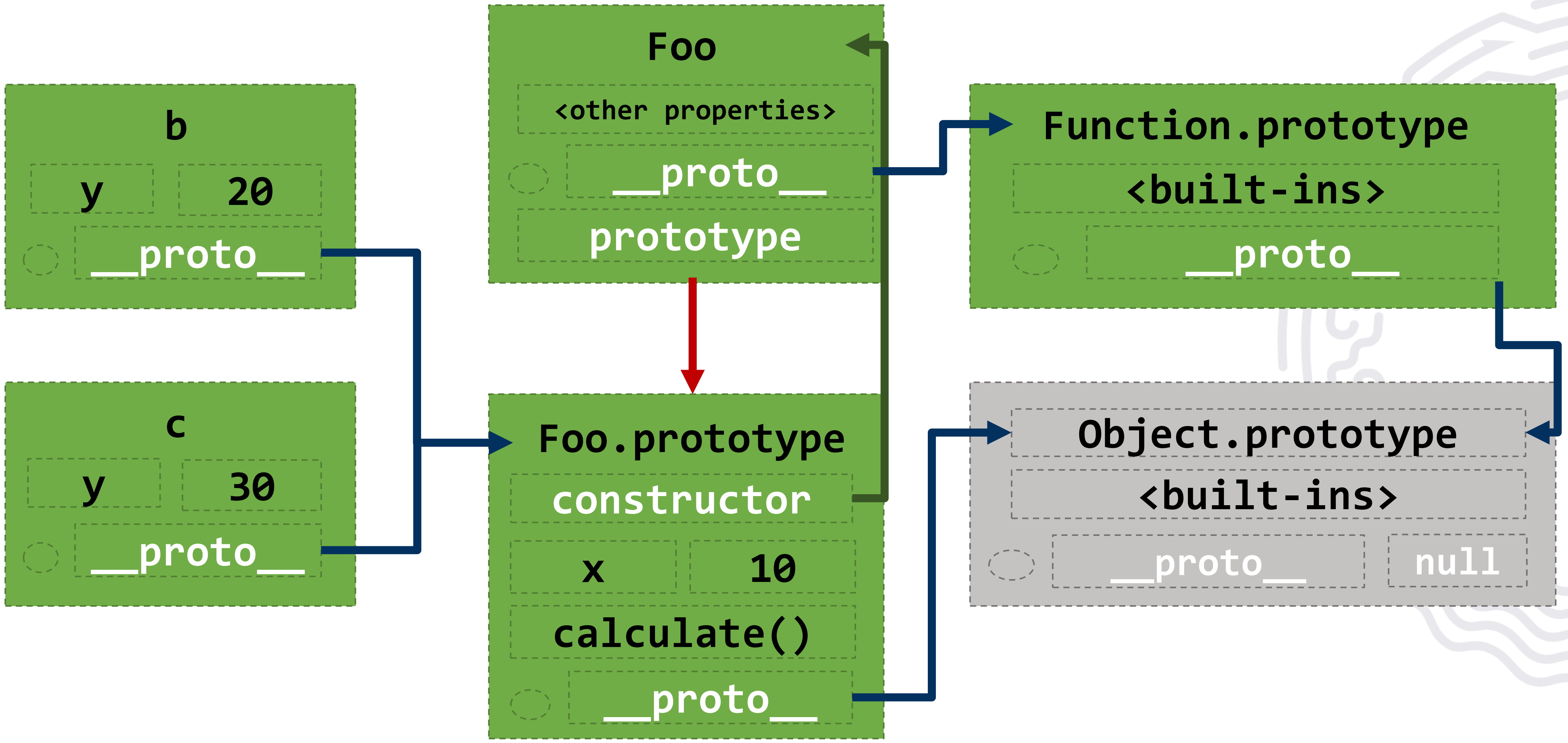
Prototype Chain - Simple Example

```
function Foo(y) {
    this.y = y;
}

Foo.prototype.x = 10;
Foo.prototype.calculate = function (z) {
    return this.x + this.y + z;
};

let b = new Foo(20);
```

Prototype Chain Diagram



Problem: Extending Prototype

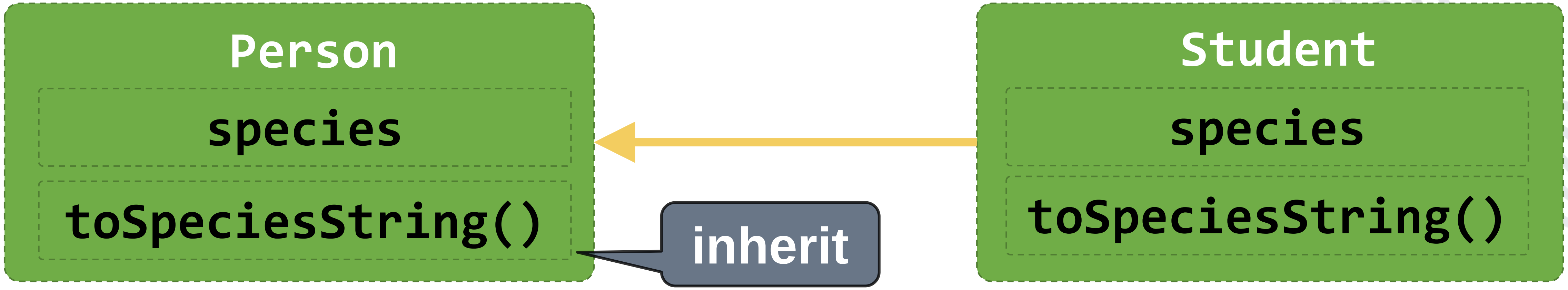
- Extend a passed class's **`prototype`** with a property **`species`** and method **`toSpeciesString()`**:
  - **`Person.prototype.species`** - holds a string value "*Human*"
  - **`Person.prototype.toSpeciesString()`** - returns
  - "I am a `{species}. {class.toString()}`"

```
new Person("Maria", "maria@gmail.com").toSpeciesString()
// "I am a Human. Person (name: Maria, email: maria@gmail.com)"
```

Solution: Extending Prototype

```
function extendPrototype(Class) {
    Class.prototype.species = "Human";
    Class.prototype.toSpeciesString = function () {
        return `I am a ${this.species}. ${this.toString()}`;
    }
}
```

```
extendPrototype(Person);
```



Live Exercise in Class (Lab)

Practice

Summary

- Inheritance allows **extending** existing classes
  - Child class inherits **data + methods** from its parent
- Objects in JS have **prototypes**
  - Objects look for **properties** in their prototype chains
  - Prototypes form a **hierarchical chain**

Questions?

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © Kingsland University – https://kingslanduniversity.com

KINGSLAND UNIVERSITY

THANK YOU