

# CMSC335

---

## Web Application Development with JavaScript



## JavaScript6

Department of Computer Science  
University of MD, College Park

Slides material developed by Ilchul Yoon, Nelson Padua-Perez

# Object Type

---

- All objects in JavaScript are descendants of Object
- All objects have a property called **\_\_proto\_\_**
- The **\_\_proto\_\_** property points to an object (called prototype) from which properties are inherited
- Objects inherit methods and properties from Object.prototype
- The **Object.prototype.isPrototypeOf(obj)** allow us to verify whether an object has Object.prototype as their prototype
- **Object.getOwnPropertyNames(obj)** allow us to get own properties of an object
- The **Object.create()** method allow us to provide a prototype object to a newly created object
- **Prototype chain**
  - Set of objects defined by the **\_\_proto\_\_** property
  - The end of the chain is a prototype with the null value (Object.prototype.\_\_proto\_\_)
- **Example:** ObjectCreation.html

# Function Properties and Methods

---

- In JavaScript, **every function is a Function object**
- **The Function constructor (native code) creates a new Function object**
- **length** property
  - Number of parameters expected by a function
- Inside of a function two objects exists
  - **arguments**
    - » Has all the arguments passed into the function
    - » It is not an array
  - **this**
    - » Reference to the **context object** the function is operating on
    - » Allows associating functions to object until runtime
    - » You can set the **this** value using `apply()`, `call()`, or `bind()`
- **Examples:** `FuncLength.html`, `FuncArguments.html`,
- **Examples:** `FuncThis.html`, `FuncApplyCallBind.html`

# Creating objects using constructor functions

---

- **To create a custom object you can:**
  - Create a function referred to as constructor function
    - » Convention is to use an uppercase initial letter for the function's name
  - Instantiate and initialize an object using **new** and the constructor function
  - **Any function called with the new operator behaves as a constructor**; without it the function behaves as a normal function

- **Example:** ConstructorFunction.html

In the example code below object creation is not efficient as we duplicate the code for the functions (we will see a better alternative later on)

# Custom Type Definition

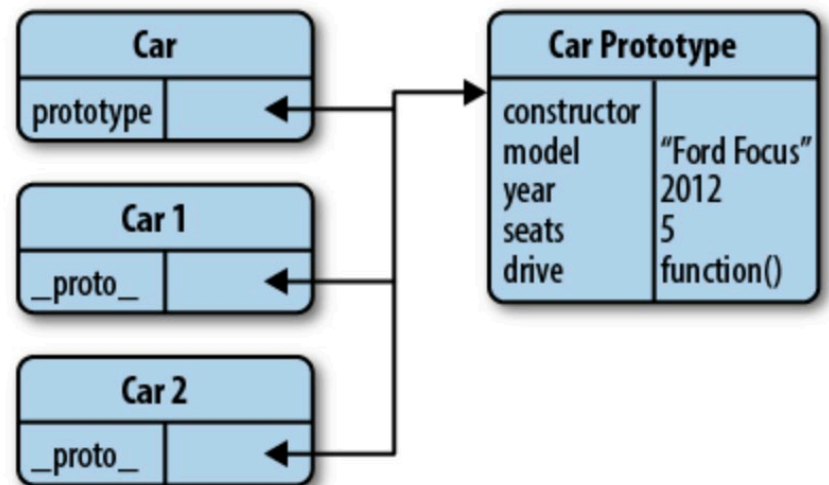
---

- Different approaches has been developed to address the creation of objects associated with a particular abstraction
  - **Constructor Pattern**
  - **Prototype Pattern**
  - **Constructor/Prototype Pattern**
- **Constructor Pattern**
  - Using constructor functions
  - Disadvantage: duplicating info object
  - The constructor function has a property called **prototype**
- **Example:** ConstructorPattern.html

# Sharing of prototype

- How **sharing of prototype** takes place when a constructor function is used to create an object using new:
  - JavaScript creates a new empty object and calls the function with “this” referring to the new object
  - The **\_\_proto\_\_** property of the new object is initialized to point to the object referred to by the **prototype** property of the constructor function
  - The new object is returned

Prototype Pattern



# Prototype Pattern

---

- The constructor function has a property called prototype
- The Constructor pattern for custom type definition has some disadvantages
  - **Each instance has its own copy of methods**
- The **Prototype** pattern addresses this problem
- **Example:** PrototypePattern.html
  - Sharing is a problem for certain properties using the Prototype Pattern

# Default Pattern for Custom Types

---

- The default pattern for custom type definition (“class definition”) combines the constructor and prototype pattern
  - **Constructor pattern defines instance variables**
  - **Prototype pattern defines common methods and properties**
- **Example:** DefaultPattern.html
  - Even if instances for an object has been created, adding a property/method to the prototype will make it immediately available

# Inheritance

---

- **Prototype chaining** : primary method for inheritance
- We can assign a particular object to the prototype property
- **Example:** Inheritance.html

# Wrap-up

---

- After this class, students should be able to:
  - Understand that every function is a Function object and has properties. (e.g., length, prototype, arguments)
  - Understand what object **this** is bound when a function is executed
  - Use call, apply, bind to use **this** as the context object in the function execution
  - Can create custom objects from construction functions
    - » Any function called with the new operator behaves as a constructor
  - Understand and implement code using constructor pattern, prototype pattern, and default pattern
  - Implement prototypical inheritance using the patterns above