**Day 40**





**“Web Development + Security”**

**Classes & Objects - Object Oriented Programming in JavaScript:**

**What is Object-Oriented Programming (OOP)?**

OOP is a programming style that organizes code into objects — reusable, logical units that hold data (properties) and behaviors (methods).

In short:  
Object = data + actions  
Example: A car object can have properties like color, brand, and methods like start() or stop().

**What is class?**

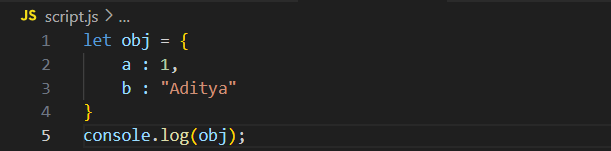
A class is a blueprint for creating objects.

A basic example of creating an object:

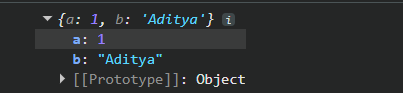
Index.html:



Script.js:



Console:

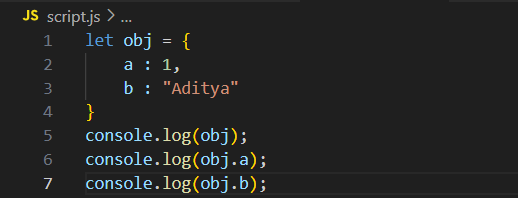


This is a JavaScript object — {a: 1, b: 'Aditya'} — which stores data in key–value pairs. Here, the key a has the value 1, and b has the value 'Aditya'. The part [[Prototype]]: Object is shown in the browser console (like Chrome DevTools) and means this object automatically inherits properties and methods from JavaScript’s built-in Object prototype (for example, methods like toString() or hasOwnProperty()).

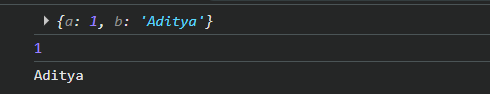
Accessing the object: we will be using the “.” As obj.a and obj.b as shown below.

Index.html: same as above

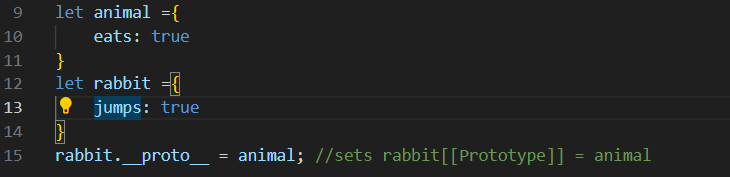
Script.js:

\

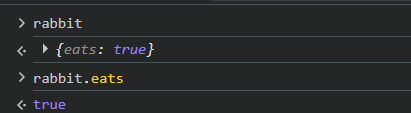
Console:



Also, we can use the prototype to assign the properties of one object to the another:

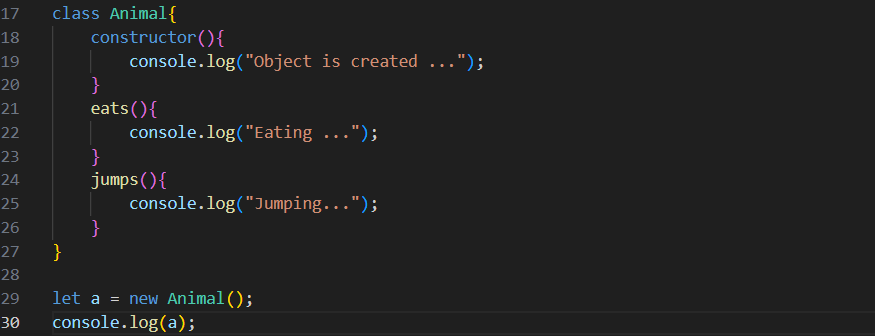
Script.js:  


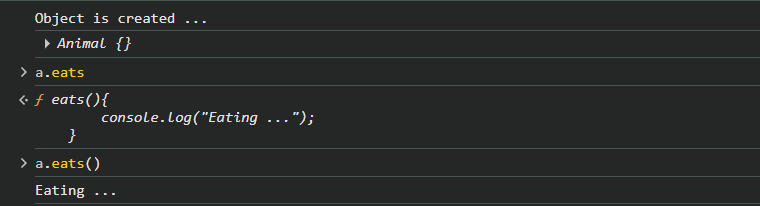
Console:



Now, creating the object using the class:

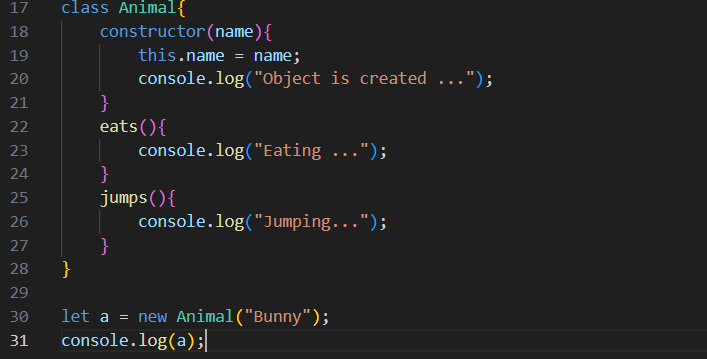
Script.js:

Console:

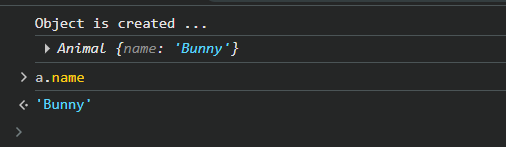


Now, creating properties in an object: we will use “this” keyword.

Script.js:

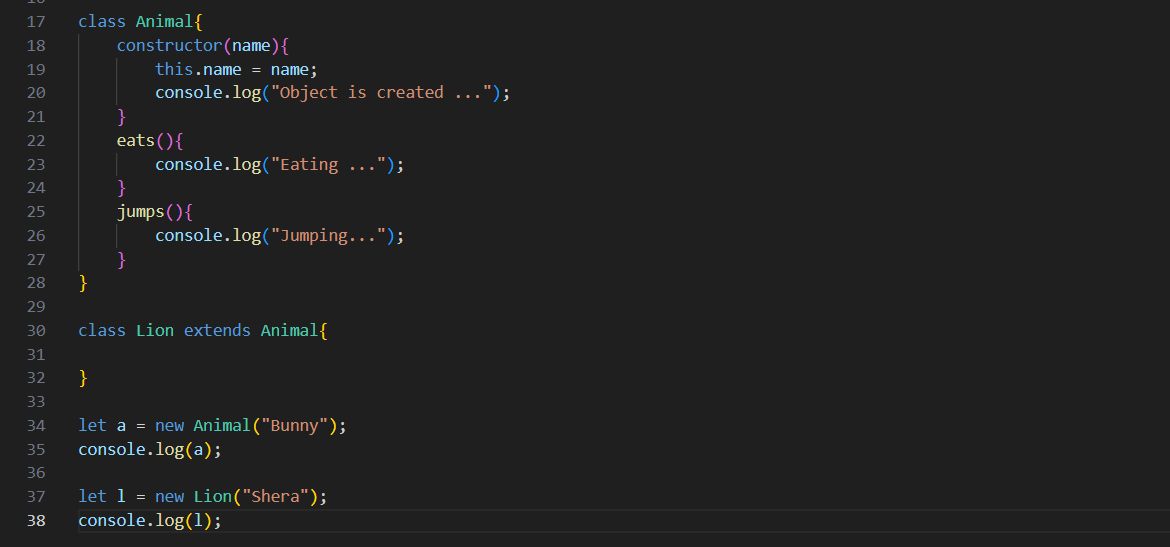


Console:

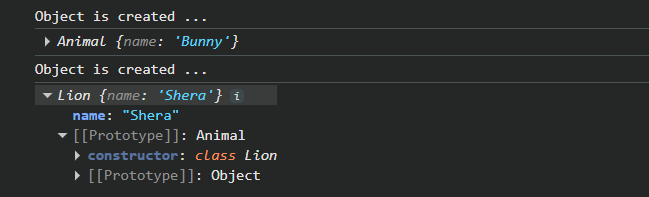


Now, using the “extend” keyword to pass the properties of Animal to the other class:

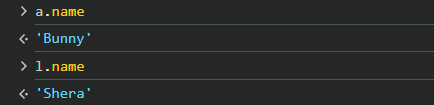
Script.js:



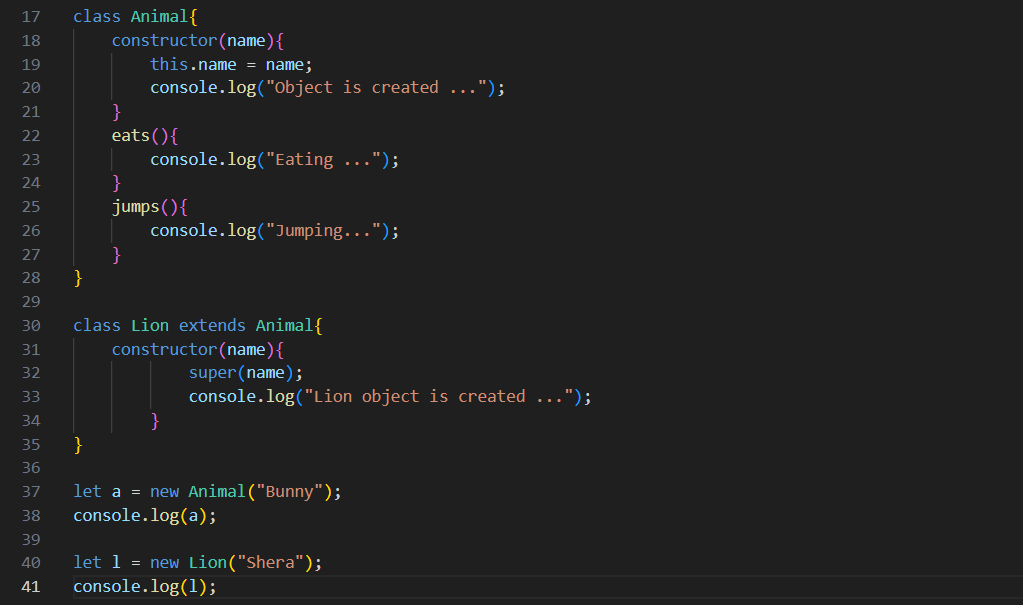
Console: clearly, prototype of Lion says it belongs to Animal.



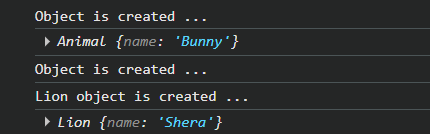
For above:



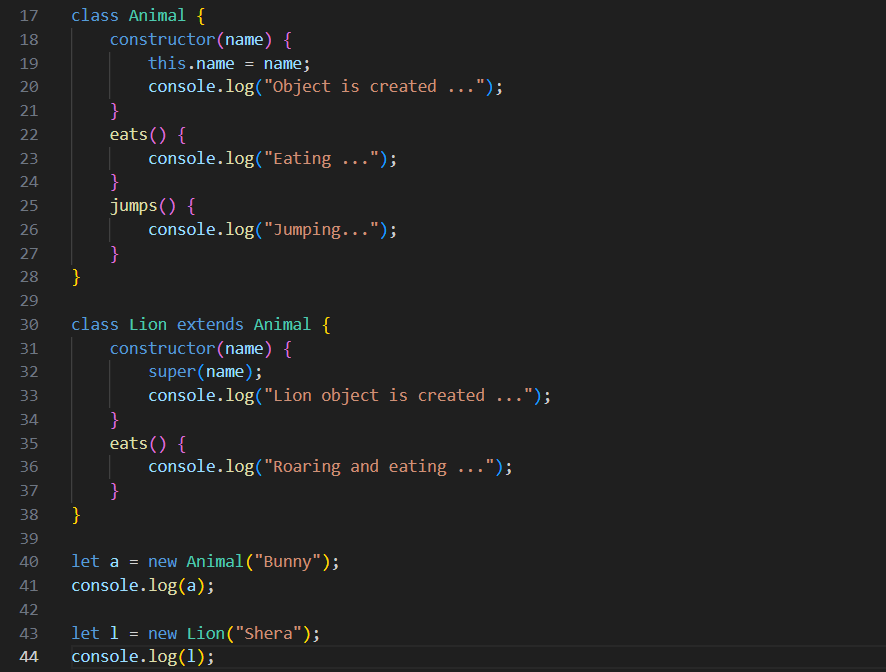
Now, giving properties to the Lion class: we will use super() keyword to achieve so.



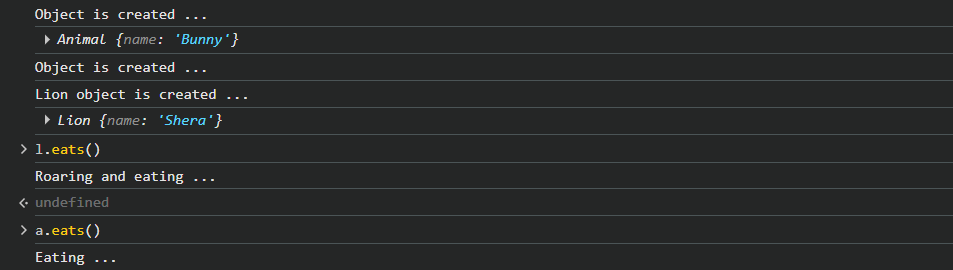
Console:



Now, overriding:



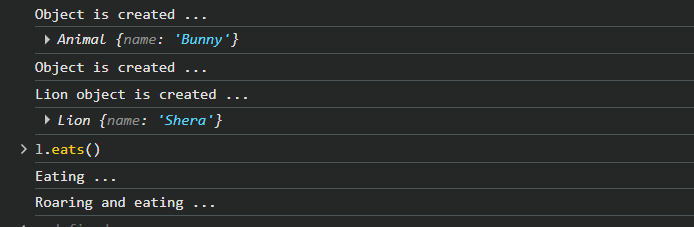
Console:



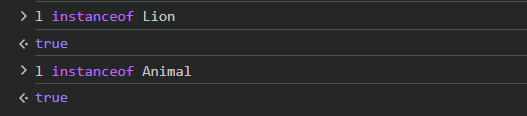
Now, what if we want the eats() of the parent class to run too? We will use the super keyword along with the method name:

Script.js:  


Console:



Now, we have “instanceof”:



--The End--