Prototype Inheritance In JavaScript | JavaScript Tutorial In Hindi #30

In the previous tutorial, we went over what object is, how to create an object, and how to access and modify object properties. Now in this tutorial, we will learn how prototypes can be used to extend objects. As we know that JavaScript is a **prototype-based language**. The object's properties and methods can be shared through generalized objects that can be cloned and extended. This sharing is known as ***prototypical inheritance.***

Every object in JavaScript has an internal property called prototype. Here is an empty object i.e., let a = {}. In this way, we normally create an object, but note that another way to accomplish this is with the object constructor: let a = new Object(). We have discussed the object in the ***tutorial#27***. Before understanding the prototype inheritance, let us first understand what inheritance is.

Inheritance:-

Inheritance let the objects to share each other's properties. In other words, Inheritance is the process by which one object can be based on another.

Prototype Inheritance:

When we try to access a property or any object's method, the JavaScript will first search on the object itself, and then it will search the object's prototype if not found. If, after checking both the object and its prototype still no match is found, JavaScript will check the linked object prototype and continue searching until the end of the prototype chain is reached. Object.prototype is at the end of the prototype chain. All the objects inherit the properties and methods of Object. When we try to search beyond the end of the chain, results in null. There are different ways to create an object in JavaScript.

- Object literal
- Function constructor
- The create() method

We have already discussed the object literal and function constructor in ***tutorial#27***. Today we will discuss **object.create()** method.

Object.create():-

The **Object.create()** method using an existing object as the prototype, creates a new object. The syntax is:

```
Object.create(proto, [propertiesObject])
```

Here is an example:

Copy

```
const myDetail= {
  isHuman: true,
  printIntro: function() {
  console.log(`My name is ${this.name}. Am I human? ${this.isHuman}`);
}};
const myself = Object.create(myDetail);
myself.name = 'Harry'; // "name" is a property set on "me", but not on "person"
myself.printIntro();
// expected output: "My name is Harry. Am I human? true"
```

To create inheritance between function constructors, call the parent constructor using call or link the prototype of the child constructor to the parent constructor prototype.

Using call() to chain constructors for an object:-

The call() allows the method or function belonging to one object to be assigned and called for another object.This method provides a new value of this to the method or function. With call(), we can write a method once and then inherit it in another object without rewriting the new object's method.
For example, the constructor for the Factory object is defined with two parameters: name and location. Other functions, Food, invoke Factory, passing this, name, and location. Factory initializes the property's name and location; the specialized functions define the category.

```
function Factory (name, location) {
  this.name = name;
  this.location = location;
}
function Food(name, location) {
 Factory.call(this, name, location);
 this.category = 'food';
}
const myFood = new Food('Nestle', 'UK' );
```

Manually set the constructor:-

The **constructor** property is used to return a reference to the object constructor function that created the instance object. We mostly use this property to define a **function-constructor** function by calling it with **a new** and prototype-inherits chain. Here is a simple example of how to set the constructor manually:

```
function Factory() {
/* ... */
 }
Factory.prototype.FactoryMethod = function FactoryMethod() {}
function Product1() {
   Factory.call(this)
}
Product1.prototype.constructor = Product1
```

Website.html code as described/written in the video

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <div class="container">
        <h1 id="heading" class='yourhead rhia is'> Welcome to Code With Harry</h1>
        <div id="myfirst" class="child red good" id="first">child 1

            <ul class="this" id='myul'>
                <li class="childul" id='fui'>this</li>
                <li class="childul">is</li>
                <li class="childul">a</li>
                <li class="childul">list </li>
                <li class="childul" id='lui'>of my dreams</li>
        </div>
        <div class="child">child 2</div>
        <div class="child red">child 3</div>
        <div class="child">child 4</div>
        <form action="none.html" method="post">
            <a href="//codewithharry.com">Go to Code With Harry</a>
            <br>
```

```
            <br>
            Search this website: <input type="text" name="Hello" id="">
            <button id="btn">Submit form</button>
            <!-- <input type="button" id='btn' value="submit"> -->
        </form>
    </div>
    <br>
    <div class="no">this is a dummy div1</div>
    <div class="no">this is a dummy div2</div>
    <div class="no">this is a dummy div3</div>
</body>
<!-- <script src="js/tut12.js"></script> -->
<!-- <script src="js/tut14.js"></script> -->
<!-- <script src="js/tut15.js"></script> -->
<!-- <script src="js/tut16.js"></script> -->
<!-- <script src="js/tut17.js"></script> -->
<!-- <script src="js/tut18.js"></script> -->
<!-- <script src="js/tut20.js"></script> -->
<!-- <script src="js/tut21.js"></script> -->
<!-- <script src="js/tut23.js"></script> -->
<!-- <script src="js/tut24.js"></script> -->
<!-- <script src="js/tut25.js"></script> -->
<!-- <script src="js/tut27.js"></script> -->
<!-- <script src="js/tut28.js"></script> -->
<script src="js/tut30.js"></script>
</html>
```

JavaScript code as described/written in the video

```
console.log("This is tutorial 30");

const proto = {
    slogan: function () {
        return `This company is the best`;
    },
    changeName: function (newName) {
        this.name = newName
    }
```

```
}

// This creates harry object
let harry = Object.create(proto);
harry.name = "harry";
harry.role = "Programmer";
// harry.changeName("Harry2")
// console.log(harry)

// This also creates harry object
const harry1 = Object.create(proto, {
    name: { value: "harry", writable: true },
    role: { value: "Programmer" },
});
harry1.changeName("Harry2")
// console.log(harry1)


// Employee constructor
function Employee(name, salary, experience) {
    this.name = name;
    this.salary = salary;
    this.experience = experience;
}

// Slogan
Employee.prototype.slogan = function () {
    return `This company is the best. Regards, ${this.name}`;
}

// Create an object from this constructor now
let harryObj = new Employee("Harry", 345099, 87);
console.log(harryObj.slogan())

// Programmer
function Programmer(name, salary, experience, language) {
    Employee.call(this, name, salary, experience);
    this.language = language;
```

```
}

// Inherit the prototype
Programmer.prototype = Object.create(Employee.prototype);

// Manually set the constructor
Programmer.prototype.constructor = Programmer;

let rohan = new Programmer("Rohan", 2, 0, "Javascript");
console.log(rohan);
```

Previous Next

CodeWithHarry

Copyright © 2022 CodeWithHarry.com