Object Prototype In javascript | JavaScript Tutorial In Hindi #28

Object Prototype In javascript | JavaScript Tutorial In Hindi #28

In this tutorial, we will discuss what prototypes are and how they help JavaScript achieve the concepts of Object-Oriented Programming. In the previous tutorial, we had learned various ways of creating objects in JavaScript. One way is to create an object in JavaScript is using the constructor function.

Introduction to JavaScript prototype:-

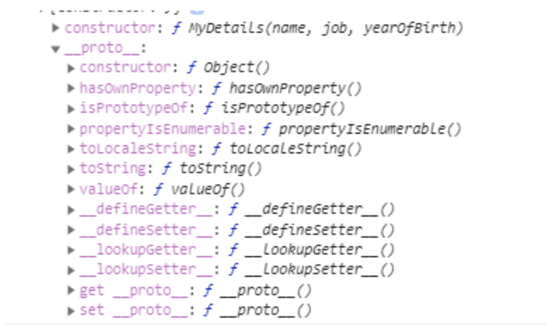
As we know that all objects in JavaScript are instances of Object. A typical object inherits properties from Object.prototype. The **Object.prototype** object has many built-in methods and properties such as toString(), valueOf(), etc.All objects see the object prototype object changes through prototype chaining unless the properties and methods subject to those changes are overridden further along the prototype chain. This is a very dangerous mechanism to override or extend object behavior.

JavaScript is a prototype-based language. Whenever we create a function using JavaScript, the JavaScript engine adds a *prototype* property inside a function. The prototype **property** is an object where we can attach methods and properties in a prototype object, which enables all the other objects to inherit these methods and properties. As we have studied in tutorial#27, one way to create an object is by using a function constructor.

```
function MyDetails(name, job, yearOfBirth){
    this.name= name;
    this.job= job;
    this.yearOfBirth= yearOfBirth;
}
console.log(MyDetails.prototype)
```

Output:-

```
▾{constructor: f}
```



```
▸ constructor: ƒ MyDetails(name, job, yearOfBirth)
▾ __proto__:
  ▸ constructor: ƒ Object()
  ▸ hasOwnProperty: ƒ hasOwnProperty()
  ▸ isPrototypeOf: ƒ isPrototypeOf()
  ▸ propertyIsEnumerable: ƒ propertyIsEnumerable()
  ▸ toLocaleString: ƒ toLocaleString()
  ▸ toString: ƒ toString()
  ▸ valueOf: ƒ valueOf()
  ▸ __defineGetter__: ƒ __defineGetter__()
  ▸ __defineSetter__: ƒ __defineSetter__()
  ▸ __lookupGetter__: ƒ __lookupGetter__()
  ▸ __lookupSetter__: ƒ __lookupSetter__()
  ▸ get __proto__: ƒ __proto__()
  ▸ set __proto__: ƒ __proto__()
```

The prototype object includes many properties and methods. Here is the explanation of a few prototype object methods.

Methods and Description:

- **hasOwnProperty():** It will return a boolean indicating whether an object contains the specified property as a direct property of that object and not inherited through the prototype chain.
- **isPrototypeOf():** It will return a boolean indicating whether the specified object is in the prototype chain of the object, this method is called upon.
- **propertyIsEnumerable():** It will return a boolean that indicates whether the specified property is enumerable or not.
- **toLocaleString():** It will return the string in the local format.
- **toString():** It will return the string.
- **valueOf()** :It will return the primitive value of the specified object.

Use of Prototype:-

JavaScript is using the prototype object in two things. First, one is to find properties and methods of an object, and the other is to implement inheritance in JavaScript. Here is an example:

f

```
function Student() {
this.name = 'Harry';
this.gender = 'Male';
}
Student.prototype.sayHi = function(){
console.log("Hello World!");
};
let std = new Student();
std.toString();
```

In the above code, the JavaScript engine checks whether the toString() method is attached to std or not. If it does not find there, it uses std _proto__ link, which points to the prototype object of Student function. If it still cannot find it there, then it goes up in the hierarchy and check the prototype object of Object function because all the objects are derived from Object in JavaScript, and look for the toString() method. Thus, it finds the toString() method in the prototype object of

Object function and so we can call std.toString().

Summary:-

This tutorial has covered JavaScript object prototypes, the prototype property, and how it can add methods to constructors, and other related topics. In the next tutorial, we will look at how we can implement inheritance functionality in JavaScript.

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>
</html>
```

JavaScript code as described/written in the video

Copy

```
console.log("This is tutorial 28");

// Object literal : Object.prototype
let obj = {
    name: "harry",
    channel: "Code With Harry",
    address: "Mars"
}

function Obj(givenName){
    this.name = givenName
}

Obj.prototype.getName = function (){
    return this.name;
}

Obj.prototype.setName = function (newName){
  this.name = newName;
}

let obj2 = new Obj("Rohan Das");
console.log(obj2);
```

Previous Next