Prototype in JavaScript

JavaScript is a dynamic language. You can attach new properties to an object at any time as shown below.

function Student() {

this.name = 'John';

this.gender = 'Male';

}

var studObj1 = new Student();

studObj1.age = 15;

alert(studObj1.age); // 15

var studObj2 = new Student();

alert(studObj2.age); // undefined

As you can see in the above example, age property is attached to studObj1 instance. However, studObj2 instance will not have age property because it is defined only on studObj1 instance.

So what to do if we want to add new properties at later stage to a function which will be shared across all the instances?

The answer is **Prototype**.

The prototype is an object that is associated with every functions and objects by default in JavaScript, where function's prototype property is accessible and modifiable and object's prototype property (aka attribute) is not visible.

Every function includes prototype object by default.

[](https://www.tutorialsteacher.com/Content/images/oo-js/prototype-1.png)

The prototype object is special type of enumerable object to which additional properties can be attached to it which will be shared across all the instances of it's constructor function.

So, use prototype property of a function in the above example in order to have age properties across all the objects as shown below.

function Student() {

this.name = 'John';

this.gender = 'M';

}

Student.prototype.age = 15;

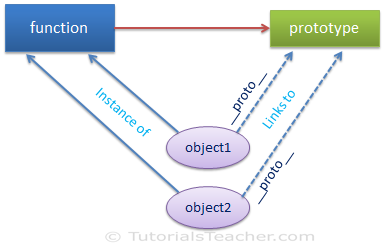
var studObj1 = new Student();

alert(studObj1.age); // 15

var studObj2 = new Student();

alert(studObj2.age); // 15

Every object which is created using literal syntax or constructor syntax with the new keyword, includes \_\_proto\_\_ property that points to prototype object of a function that created this object.

[](https://www.tutorialsteacher.com/Content/images/oo-js/prototype-2.png)

function Student() {

this.name = 'John';

this.gender = 'M';

}

var studObj = new Student();

console.log(Student.prototype); // object

console.log(studObj.prototype); // undefined

console.log(studObj.\_\_proto\_\_); // object

console.log(typeof Student.prototype); // object

console.log(typeof studObj.\_\_proto\_\_); // object

console.log(Student.prototype === studObj.\_\_proto\_\_ ); // true