A string object and a string value are not the same thing:

js> p = "Foo"

Foo

js> p.weight = 42

42

js> p.weight // Returns undefined

js> q = new String("Foo")

Foo

js> q.weight = 42

42

js> q.weight

42

The string value cannot have new properties. The same thing is valid for other types.

What is going on here that an string is not an object? Am i confusing JavaScript with some other languages, where everything is an object?

*"Everything is an object"*... that's one of the big misconceptions that exist all around the language.

**Not** *everything* is an object, there are what we call *primitive values*, which are string, number, Boolean, null, and undefined.

That's true, a string is a **primitive value**, but you can access all the methods inherited from String.prototype as if it were an object.

The property [accessor operators](http://ecma262-5.com/ELS5_HTML.htm#Section_11.2.1) (the dot and the bracket notation), temporarily convert the string value to a String object, for being able to access those methods, e.g.:

"ab".charAt(1); // "b"

What happens behind the scenes is something like this:

new String("ab").charAt(1); // "b", temporal conversion ToObject

As with the other primitive values, such as Boolean, and Number, there are *object wrappers*, which are simply objects that contain the primitive value, as in your example:

var strObj = new String("");

strObj.prop = "foo";

typeof strObj; // "object"

typeof strObj.prop; // "string"

While with a primitive:

var strValue = "";

strValue.prop = "foo";

typeof strValue; // "string"

typeof strValue.prop; // "undefined"

And this happens because again, the property accessor on the second line above, creates a new temporal object, as:

var strValue = "";

new String(strValue).prop = "foo"; // a new object which is discarded

//...