

Javascript Prototype Behaviour in PHP

Monday, September 29th 2008

One of the "neat" things in Javascript is you are able to dynamically add or change methods of a class and automatically update every instance of that class. Some of the things I usually find useful are adding to the String class, like so:

```
String.prototype.htmlSpecialChars = function() {
    return this.replace(/</g, '&#060;').replace(/>/g, '&#062;');
}
String.prototype.trim = function() {
    return this.replace(/^[s+]|s+$/g, '');
}
```

Obviously we cannot do this in PHP, and why would we, right? However we can emulate this behaviour to a certain extent using my "neat" little [Prototype](#) class. With this Prototype class we can dynamically add properties and methods to any class, and they will be inherited by all instances of that class.

Let's look at the following "normal" PHP code.

```
class Person extends Prototype
{
    public $name;
    public $gender;

    public function gender()
    {
        printf("%s is %sn", $this->name, $this->gender);
    }
}

$matt = new Person;
$matt->name = 'Matt';
$matt->gender = 'male';
$matt->gender();

// Matt is male
```

Now, there is nothing magical or out-of-the-ordinary going on here. We just instantiate the Person class and setup some properties. Calling the `gender()` method outputs a nice little string for us.

However, you see that the Person class is actually a child of the Prototype class. This will allow us to do some

of that "neat" Javascript stuff. Using Prototype, let us expand the Person class to add an \$age property and an age() method to output a nice string. Like so:

```
Person::add_property('age');
Person::add_method('age', 'printf("%s is a %d year old %sn", $this->name,
$this->age, $this->gender);');

$matt->age = 28;
$matt->age();

// Matt is a 28 year old male
```

Now all instances of Person inherit the \$age property and age() method. So we can create a new Person, Susie, and this object will now have the age stuff.

```
$susie = new Person;
$susie->name = 'Susie';
$susie->gender = 'female';
$susie->age = 21;
$susie->age();

// Susie is a 21 year old female
```

One limitation of the Prototype class though, is you cannot overload a current method. So the following code, that attempts to overload the gender() method, will not work.

```
Person::add_method('gender', 'printf("%s is a %d year old %sn", $this->name,
$this->age, $this->gender);');

$matt->gender();

// Matt is male
```

There are also many, many, many other problems with this Prototype class. Some of which are:

- Â» The '\$this' keyword is reserved, so it actually does a string replace and uses '\$self' instead.
- Â» You cannot access/add new methods or properties statically (until PHP 5.3 with __callStatic()).
- Â» It uses create_function, so every "method" is actually defined in the global namespace.
- Â» Iteration does not work, although it could possibly be done with Iterator, Countable, et al.
- Â» You cannot reference static variables/methods in your add method.

Â» You cannot share methods between classes.

Â» And so on and so forth...

This class was just an experiment to see if it was at all possible to implement something like Javascript's prototype behaviour in PHP with out using the [Runkit PECL extension](#). I had no intention of actually making this usable in production, for many reasons , although it was fun. If you have any improvements or additions to Prototype please add them to the [wiki page](#), or paste them in comment.