Basic JavaScript Part 4: Enforcing New on Constructor Functions

December 21st, 2010

As this is already the fourth blog post using the "Basic JavaScript" theme, I guess we're slowly getting a small blog series on our hands. Here are the links to the previous installments:

1. Functions
2. Objects
3. Prototypes

In the blog post on objects, I mentioned that there's a general naming convention for constructor functions, using Pascal casing as opposed to the usual Camel case naming style for JavaScript functions. When following this naming convention we can make a visual distinction between a constructor function and a normal function. We want to make this distinction because we always need to call a constructor function with the *new* operator.

```
function Podcast() {
    this.title = 'Astronomy Cast';
    this.description = 'A fact-based journey through the galaxy.';
    this.link = 'http://www.astronomycast.com';
}

Podcast.prototype.toString = function() {
   return 'Title: ' + this.title;
};

var podcast = new Podcast();
podcast.toString();
```

Suppose that for some reason this naming convention slips our mind and we forget using the*new* operator when calling this constructor function. This usually leads to some nasty and unexpected behavior that is sometimes very hard to track down. What actually happens when we omit the *new* keyword, is that *this* now points to the global object (the *window* object when the JavaScript code is running in the browser) instead of the object that we intended to create. As a result, the properties in the constructor function are now added to the global object. This is definitely not what we want.

Rather than relying purely on a naming convention, we might want to enforce that every time a constructor function is called, this function is invoked properly using the *new* operator. In order to achieve this, we can add the following check to the beginning of the constructor function shown earlier:

```
function Podcast() {
    if(false === (this instanceof Podcast)) {
        return new Podcast();
    }

    this.title = 'Astronomy Cast';
    this.description = 'A fact-based journey through the galaxy.';
    this.link = 'http://www.astronomycast.com';
}

Podcast.prototype.toString = function() {
   return 'Title: ' + this.title;
};

var podcast1 = Podcast();
var podcast2 = new Podcast();
```

Adding this check verifies whether *this* references an object created by our constructor function, and if not, the constructor function is called again but this time using the *new*operator.

So adding this check to all your constructor functions guarantees that these are invoked correctly using *new*.

Till next time.