Prototype Based State Manager

Posted At : April 6, 2008 5:37 PM | Posted By : Jon Hartmann
Related Categories: Javascript

I've known that you could use anchor tags as a way to manage the state of a Javascript application for a while now, but I've never really messed with how that is actually setup. I got bored on Friday, and this is what I came up with.

I've no idea if this is the way that pros at JS would design this kind of functionality, but I think its useful enough.

On second thought, this would probably be best done as a singleton, since having multiple instances of the StateManager could cause conflict. Adding a PeriodicExecutor to monitor the state for changes might also be an option, as well as adding a method to allow the registering of more then one listener function.

This solution requires the use of the **Prototype Library**, version 1.6.

```
<InvalidTag>
var StateManager = Class.create({
    initialize: function (config) {
        this._config = config ? config : {};
            this._config.defaultState = this._config.defaultState ? this._config.defaultState :
'default';

        this._state = document.URL.split('#')[1] ? document.URL.split('#')[1] :
this._config.defaultState;

        if ( this._config.onStateChange ) {
            this._config.onStateChange(this._state);
        }
    },
    setState: function (state) {
        if ( this._state != state ) {
            this._state = state;
            parent.location = document.URL.split('#')[0] + '#' + this._state;
            if ( this._config.onStateChange ) {
                this._config.onStateChange(this._state);
            }
        }
    },
    getState: function () {
        if ( this._state == undefined ) {
            return this._config.defaultState;
        } else {
            return this._state;
        }
    }
});
</script>
```

Using the code is simple, just instantiate a new StateManager and give it a default event handler:

```
function handleChange (state) {
    alert(state);
}
```

```
var StateControl = new StateManager({
    onStateChange : handleChange
});
```

If you need to change the event, just link it to a button or event:

```
<button onClick="StateController.setState('newState');">New State</button>
```