SafeBox-Lite v0.0.2 A Javascript Library

[With In-built Cache]

SafeBox-Lite is a Javascript library which helps you to run your Javascript function in isolated execution context. It has following features which helps the programmer to run their code-snippet and plugin in safer way.

- Provide Isolated Execution.
- Everything in local variable; a perfect GC friendly.
- In-built internal soft cache gives safer place to store code-snippet with performance.
- A better place to keep your class/constructor with flexible portability.

And powerful feature of SafeBox -Lite is, it will restore the default data-type while executing the function. It will protect your module from any anonymous global overridden. You can see this in following example.

```
var Array = null;
window.Object = null;
SafeBox(function mySaferFunction() {
        var myArray = new Array(1, 2);
        var myObject = Object.create({
                foo : "bar"
        });
        console.log("myArray values are", myArray);
        console.log("myObject values are", myObject);
})();
```

The second powerful feature of SafeBox-Lite is, it will force Javascript developer to run the function with local variables. It is so flexible to the Garbage collection(GC). And vice versa, any changes in function within SafeBox scope will not affect outside scope. You can see this in following example.

```
(function() {
        var mainScopeVar = "Hello World!";
                SafeBox(function(){
                console.log(mainScopeVar); // mainScopeVar is not defined
        })();
})();
```

Alternatively you can run the function as follows,

```
(function() {
        var mainScopeVar = "Hello World!";
        SafeBox(function(mainScopeVar) {
                console.log(mainScopeVar);// Prints "Hello World!"
        })(mainScopeVar);
```

```
})();
```

The third powerful feature is, It is doing seem-less internal soft cache on any named Javascript function and also hide them within its closure context. Based on this, SafeBox-Lite ensures the performance of your program.

All of these comes together and make SafeBox-Lite as a perfect place to hold your Class or Constructor functions.

The fourth most important thing is, SafeBox-Lite supports one time compilation of your classes. This compiled class won't be affect any further overridden in SafeBox-Lite context. So SafeBox-Lite stands on the line of changing object, not a Class definition.

You can see this in following example,

```
var Person = SafeBox(function Person() {
        function Person(name, age) {
                this.name = name;
                this.age = age;
        };
        Person.prototype.display = function() {
                console.log("Person name is " + this.name);
        };
        return Person;
})();

// Attempt to rewrite  class definition
var Person = SafeBox (function Person () {
        function Person(name, age) {
                this.name = "Overriden " + name;
                this.age = age;
        };

        Person.prototype.display = function() {
                console.log("Person name is " + this.name);
        };
        return Person;
})();

var person2 = new Person("foo", 27);
console.log('Person 2', person2);
```

And you can restore any class definition by simply passing the name of class. Since SafeBox-Lite attached with "window" object, you can port your class definition anywhere across your project without create any global object. It prevents the global window object from pollution. You can keep your window object with less weight. You can do this as follows,

```
(function() {
        // I am going to create "person" object and I assume that "Person" class
        // already digested by SafeBox-Lite.
        var Person = SafeBox("Person")();

        // Creating "person" object
        var person = new Person();
})();
```