STACK

Procedural - Tight Coupling

```
<html>
    <head>
        <title>Procedural Stack with Tight Coupling</title>
    <script

src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.2/jquery.mi
n.js">
    </script>
    </head>
    <body>
        <div>Procedural Stack with Tight Coupling</div>
        <div id="rpisesetup">
         <textarea id="rpiseconsole"></textarea>
         <script
src="https://sites.google.com/a/rajeshpatkar.com/library/hub/rpis
econsole.js">
         </script>
        </div>
        <script>

            var st1 = 10;
            var stk1 = new Array(10);

            function push1(v) {
                if (st1 === 0) {
                    rpiseconsole.log("Stack Overflow");
                }
                else {
                    st1 = st1 - 1;
                    stk1[st1] = v;
                }
            }

            function pop1() {
```

```
    if (st1 === 10) {
        rpiseconsole.log("Stack Underflow");
    }
    else {
        var temp = stk1[st1];
        st1 = st1 + 1;
        return temp;
    }
}

function print1() {
    rpiseconsole.log("Printing Stack");
    for (var i = st1 ; i < 10; i++) {
        rpiseconsole.log(stk1[i]);
    }
};


push1(10);
push1(20);
push1(30);
print1();

var st2 = 10;
var stk2 = new Array(10);

function push2(v) {
    if (st2 === 0) {
        rpiseconsole.log("Stack Overflow");
    }
    else {
        st2 = st2 - 1;
        stk2[st2] = v;
    }
}

function pop2() {
    if (st2 === 10) {
        rpiseconsole.log("Stack Underflow");
    }
    else {
        var temp = stk2[st2];
        st2 = st2 + 1;
        return temp;
```

```
                return temp;
            }
        }

        function print2() {
            rpiseconsole.log("Printing Stack");
            for (var i = st2; i < 10; i++) {
                rpiseconsole.log(stk2[i]);
            }
        };

        push2(22);
        push2(33);
        push2(44);
        print2();
        var v = pop2();

        rpiseconsole.log("Last Value popped in s2 is " + v);
        print2();

    </script>
</body>
</html>
```

Procedural - Loose Coupling

```
<!DOCTYPE html>
<html>
    <head>
        <title>Procedural Stack with Loose Coupling</title>
        <script

src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.2/jquery.mi
n.js">
        </script>
    </head>
    <body>

        <div>Procedural Stack with Loose Coupling</div>
        <div id="rpisesetup">
         <textarea id="rpiseconsole"></textarea>
         <script
src="https://sites.google.com/a/rajeshpatkar.com/library/hub/rpis
econsole.js">
```

```
nsole.js">
    </script>
</div>
<script>

    function push(st, stk, v) {
        if (st[0] === 0) {
            rpiseconsole.log("Stack Overflow");
        }
        else {
            st[0] = st[0] - 1;
            stk[st[0]] = v;
        }
    }

    function pop(st,stk) {
        if (st[0] === 10) {
            rpiserpiseconsole.log("Stack Underflow");
        }
        else {
            var temp = stk[st[0]];
            st[0] = st[0] + 1;
            return temp;
        }
    }

    function print(st,stk) {
        rpiseconsole.log("Printing Stack");
        for (var i = st[0] ; i < 10; i++) {
            rpiseconsole.log(stk[i]);
        }
    };


    var st1 = [10];
    var stk1 = new Array(10);

    push(st1,stk1,10);
    push(st1,stk1,20);
    push(st1,stk1,30);
    print(st1,stk1);

    var st2 = [10];
    var stk2 = new Array(10);
```

```
            push(st2,stk2,22);
            push(st2,stk2,33);
            push(st2,stk2,44);
            print(st2,stk2);
            var v = pop(st2,stk2);

            rpiseconsole.log("Last Value popped in s2 is " + v);
            print(st2,stk2);
        </script>
    </body>
</html>
```

Procedural - Composite Variable

```
<html>
    <head>
        <title>Procedural Stack with Composite Variable</title>
        <script

src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.2/jquery.mi
n.js">
        </script>
    </head>
    <body>

        <div>Procedural Stack with Composite Variable</div>
        <div id="rpisesetup">
         <textarea id="rpiseconsole"></textarea>
         <script
src="https://sites.google.com/a/rajeshpatkar.com/library/hub/rpis
econsole.js">
         </script>
        </div>
        <script>

            function push(s , v) {
                if (s.st === 0) {
                    rpiseconsole.log("Stack Overflow");
                }
                else {
                    s.st s.st - 1;
                    s.stk[s.st] = v;
```

```
    }
}

function pop(s) {
    if (s.st === 10) {
        rpiseconsole.log("Stack Underflow");
    }
    else {
        var temp = s.stk[s.st];
        s.st = s.st + 1;
        return temp;
    }
}

function print(s) {
    rpiseconsole.log("Printing Stack");
    for (var i = s.st ; i < 10; i++) {
        rpiseconsole.log(s.stk[i]);
    }
};


var s1 = {
    st  : 10,
    stk : new Array(10)
};

push(s1,10);
push(s1,20);
push(s1,30);
print(s1);

var s2 = {
    st  : 10,
    stk : new Array(10)
};

push(s2,22);
push(s2,33);
push(s2,44);
print(s2);
var v = pop(s2);

rpiseconsole.log("Last Value popped in s2 is " + v);
```

```
            print(s2);

        </script>
    </body>
</html>
```

Object

```
<!DOCTYPE html>
<!--
 © Rajesh Patkar Institute Of Software Engineering.
   Codeparatus by Rajesh Patkar.
-->

<html>
    <head>
        <title>Stack as Object</title>
    </head>
    <body>

        <div>Stack as Object</div>

        <script>

            function push(v) {
                if (this.st === 0) {
                    console.log("Stack Overflow");
                }
                else {
                    this.st = this.st - 1;
                    this.stk[this.st] = v;
                }
            }

            function pop() {
                if (this.st === 10) {
                    console.log("Stack Underflow");
                }
                else {
                    var temp = this.stk[this.st];
                    this.st = this.st + 1;
                    return temp;
                }
```

```
        }

        function print() {
            console.log("Printing Stack");
            for (var i = this.st ; i < 10; i++) {
                console.log(this.stk[i]);
            }
        };


        var s1 = {
            st: 10,
            stk: new Array(10),
            push: push,
            pop: pop,
            print: print
        };

        s1.push(10);
        s1.push(20);
        s1.push(30);
        s1.print();

        var s2 = {
            st: 10,
            stk: new Array(10),
            push: push,
            pop: pop,
            print: print
        };

        s2.push(22);
        s2.push(33);
        s2.push(44);
        s2.print();
        var v = s2.pop();

        console.log("Last Value popped in s2 is " + v);
        s2.print();

    </script>

</body>
```

tml>

Factory

```
<!DOCTYPE html>
<!--
 © Rajesh Patkar Institute Of Software Engineering.
   Codeparatus by Rajesh Patkar.
-->

<html>
    <head>
        <title>Stack With Factory </title>
    </head>
    <body>

        <div>Stack using Factory</div>

        <script>

            function push(v) {
                if (this.st === 0) {
                    console.log("Stack Overflow");
                }
                else {
                    this.st = this.st - 1;
                    this.stk[this.st] = v;
                }
            }

            function pop() {
                if (this.st === 10) {
                    console.log("Stack Underflow");
                }
                else {
                    var temp = this.stk[this.st];
                    this.st = this.st + 1;
                    return temp;
                }
            }

            function print() {
                console.log("Printing Stack");
```

```
            for (var i = this.st ; i < 10; i++) {
                console.log(this.stk[i]);
            }
        };


        function MyStackFactory() {
            var obj = {
               st: 10,
               stk: new Array(10),
               push: push,
               pop: pop,
               print: print
           };
           return obj;
        }

        var s1 = MyStackFactory();
        s1.push(10);
        s1.push(20);
        s1.push(30);
        s1.print();

        var s2 = MyStackFactory();
        s2.push(22);
        s2.push(33);
        s2.push(44);
        s2.print();
        var v = s2.pop();

        console.log("Last Value popped in s2 is " + v);
        s2.print();

    </script>

</body>
</html>
```

Constructor

```
<!DOCTYPE html>
<!--
 © Rajesh Patkar Institute Of Software Engineering.
```

Codeparatus by Rajesh Patkar.

```
<html>
    <head>
        <title>Stack With Constructor </title>
    </head>
    <body>

        <div>Stack using Constructor</div>

        <script>

            function push(v) {
                if (this.st === 0) {
                    console.log("Stack Overflow");
                }
                else {
                    this.st = this.st - 1;
                    this.stk[this.st] = v;
                }
            }

            function pop() {
                if (this.st === 10) {
                    console.log("Stack Underflow");
                }
                else {
                    var temp = this.stk[this.st];
                    this.st = this.st + 1;
                    return temp;
                }
            }

            function print() {
                console.log("Printing Stack");
                for (var i = this.st ; i < 10; i++) {
                    console.log(this.stk[i]);
                }
            };


            function MyStack() {
                this.st = 10;
```

```
            this.stk = new Array(10);
            this.push = push;
            this.pop = pop;
            this.print = print;
        }

        var s1 = new MyStack();
        s1.push(10);
        s1.push(20);
        s1.push(30);
        s1.print();

        var s2 = new MyStack();
        s2.push(22);
        s2.push(33);
        s2.push(44);
        s2.print();
        var v = s2.pop();

        console.log("Last Value popped in s2 is " + v);
        s2.print();

    </script>

</body>
</html>
```

Inheritance - using base

```
<!DOCTYPE html>
<!--
 © Rajesh Patkar Institute Of Software Engineering.
   Codeparatus by Rajesh Patkar.
-->

<html>
    <head>
        <title>Stack With Explicit Prototype </title>
    </head>
    <body>

        <div>Stack using Explicit Prototype</div>
```

```
<script>

    function push(v) {
        if (this.st === 0) {
            console.log("Stack Overflow");
        }
        else {
            this.st = this.st - 1;
            this.stk[this.st] = v;
        }
    }

    function pop() {
        if (this.st === 10) {
            console.log("Stack Underflow");
        }
        else {
            var temp = this.stk[this.st];
            this.st = this.st + 1;
            return temp;
        }
    }

    function print() {
        console.log("Printing Stack");
        for (var i = this.st ; i < 10; i++) {
            console.log(this.stk[i]);
        }
    };


    var base = {};
    base.push = push;
    base.pop = pop;
    base.print = print;

    function MyStack() {
        this.st = 10;
        this.stk = new Array(10);
        this.__proto__ = base;
    }

    var s1 = new MyStack();
```

```
            s1.push(10);
            s1.push(20);
            s1.push(30);
            s1.print();

            var s2 = new MyStack();
            s2.push(22);
            s2.push(33);
            s2.push(44);
            s2.print();
            var v = s2.pop();

            console.log("Last Value popped in s2 is " + v);
            s2.print();

        </script>

    </body>
</html>
```

Inheritance - using prototype

```
<!DOCTYPE html>
<!--
 © Rajesh Patkar Institute Of Software Engineering.
   Codeparatus by Rajesh Patkar.
-->

<html>
    <head>
        <title>Stack With Prototype</title>
    </head>
    <body>

        <div>Stack using Prototype</div>

        <script>

            function MyStack() {
                this.st = 10;
                this.stk = new Array(10);
            }
```

```
MyStack.prototype.push = function(v) {
    if (this.st === 0) {
        console.log("Stack Overflow");
    }
    else {
        this.st = this.st - 1;
        this.stk[this.st] = v;
    }
};

MyStack.prototype.pop = function() {
    if (this.st === 10) {
        console.log("Stack Underflow");
    }
    else {
        var temp = this.stk[this.st];
        this.st = this.st + 1;
        return temp;
    }
};

MyStack.prototype.print = function() {
    console.log("Printing Stack");
    for (var i = this.st ; i < 10; i++) {
        console.log(this.stk[i]);
    }
};

var s1 = new MyStack();
s1.push(10);
s1.push(20);
s1.push(30);
s1.print();

var s2 = new MyStack();
s2.push(22);
s2.push(33);
s2.push(44);
s2.print();
var v = s2.pop();

console.log("Last Value popped in s2 is " + v);
s2.print();
```

```
    </script>

</body>
</html>
```

Private Idiom

```
<!DOCTYPE html>
<!--
 © Rajesh Patkar Institute Of Software Engineering.
   Codeparatus by Rajesh Patkar.
-->

<html>
    <head>
        <title>Stack With Private Idiom </title>
    </head>
    <body>

        <div>Stack using Private Idiom</div>

        <script>

            function MyStack() {
                st = 10;
                stk = new Array(10);
                function push(v) {
                    if (st === 0) {
                        console.log("Stack Overflow");
                    }
                    else {
                        st = st - 1;
                        stk[st] = v;
                    }
                }

                function pop() {
                    if (st === 10) {
                        console.log("Stack Underflow");
                    }
                    else {
                        var temp = stk[st];
                        st = st + 1;
```

```
                    return temp;
                }
            }

            function print() {
                console.log("Printing Stack");
                for (var i = st; i < 10; i++) {
                    console.log(stk[i]);
                }
            }
            this.push = push;
            this.pop = pop;
            this.print = print;
        }

        var s1 = new MyStack();
        s1.push(10);
        s1.push(20);
        s1.push(30);
        s1.print();

        var s2 = new MyStack();
        s2.push(22);
        s2.push(33);
        s2.push(44);
        s2.push(55);
        s2.print();
        var v = s2.pop();

        console.log("Last Value popped in s2 is " + v);
        s2.print();

    </script>

</body>
</html>
```