Previous Next

JavaScript Closure

Closure is one of important concept in JavaScript. It is widely discussed and still confused concept. Let's understand what the closure is.

First of all, let's see the definition of the Closure given by Douglas Crockford: crockford.com/javascript/private.html

Closure means that an inner function always has access to the vars and parameters of its outer function, even after the outer function has returned.

You have learned that we can create nested functions in JavaScript. Inner function can access variables and parameters of an outer function (however, cannot access arguments object of outer function). Consider the following example.

```
function OuterFunction() {

    var outerVariable = 1;

    function InnerFunction() {
        alert(outerVariable);
```

```
    }

    InnerFunction();
}
```

In the above example, InnerFunction() can access outerVariable.

Now, as per the definition above, InnerFunction() can access outerVariable even if it will be executed separately. Consider the following example.

Example: Closure

```
function OuterFunction() {

    var outerVariable = 100;

    function InnerFunction() {
        alert(outerVariable);
    }

    return InnerFunction;
}
var innerFunc = OuterFunction();

innerFunc(); // 100
```

Try it

In the above example, `return InnerFunction;` returns InnerFunction from OuterFunction when you call OuterFunction(). A variable `innerFunc` reference the InnerFunction() only, not the OuterFunction(). So now, when you call innerFunc(), it can still access `outerVariable` which is declared in OuterFunction(). This is called Closure.

One important characteristic of closure is that outer variables can keep their states between multiple calls. Remember, inner function does not keep the separate copy of outer variables but it reference outer variables, that means value of the outer variables will be changed if you change it using inner function.

A function can return another function in JavaScript. A function which is assigned to a variable is called function expression.

Example: Closure

```
function Counter() {
    var counter = 0;

    function IncreaseCounter() {
        return counter += 1;
    };

    return IncreaseCounter;
}

var counter = Counter();
alert(counter()); // 1
alert(counter()); // 2
alert(counter()); // 3
alert(counter()); // 4
```

Try it

In the above example, outer function Counter returns the reference of inner function IncreaseCounter(). IncreaseCounter increases the outer variable counter to one. So calling inner function multiple time will increase the counter to one each time.

Closure is valid in multiple levels of inner functions.

Example: Closure

```
function Counter() {

    var counter = 0;

    setTimeout( function () {
        var innerCounter = 0;
        counter += 1;
        alert("counter = " + counter);

        setTimeout( function () {
            counter += 1;
            innerCounter += 1;
            alert("counter = " + counter + ", innerCounter = " + innerCounter)
        }, 500);

    }, 1000);
};

Counter();
```

Try it

As per the closure definition, if inner function access the variables of outer function then only it is called closure.

The following is not a closure.

```
var Counter = (function () {
        var i = 0;
        return { counter : i += 1 };
})();
```

When to use Closure?

Closure is useful in hiding implementation detail in JavaScript. In other words, it can be useful to create private variables or functions.

The following example shows how to create private functions & variable.

Example: Closure

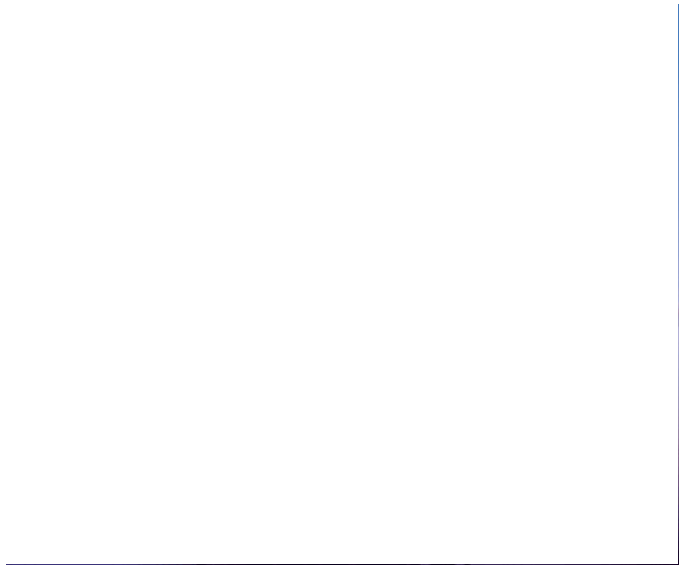
```
var counter = (function() {
  var privateCounter = 0;
  function changeBy(val) {
    privateCounter += val;
  }
  return {
    increment: function() {
      changeBy(1);
    },
    decrement: function() {
      changeBy(-1);
    },
    value: function() {
```

```
        return privateCounter;
    }
  };
})();

alert(counter.value()); // 0
counter.increment();
counter.increment();
alert(counter.value()); // 2
counter.decrement();
alert(counter.value()); // 1
```

Try it

In the above example, increment(), decrement() and value() becomes public function because they are included in the return object, whereas changeBy() function becomes private function because it is not returned and only used internally by increment() and decrement().



Previous Next