INTERPARK

2017 NOV.

{ .JS }

JavaScript Lab

자바스크립트의 변수 스코프
Variable Scope

JavaScript Lab 2

01. 전역객체 사용지양

- 공통 네임스페이스 오염
- **이름 충돌 가능성 내포**
- 구분된 요소들 간의 불필요한 결합 초래(모듈성 저하)
= 편리함으로 인해 수반되는 결과들
- 가능하면 지역변수 사용
- 전역 객체 사용시 namespace 사용 권고

예제

```
1
2   //1. 전역객체 사용 지양
3   var myApp = {}
4
5   myApp.id = 0;
6
7   myApp.next = function(){
8     return myApp.id++;
9   }
10
11  myApp.reset = function(){
12    myApp.id = 0;
13  }
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

02. 클로저(Closure)

> 내부 함수에서 부모함수의 변수 접근가능

예제

```
1
2   //2. 클로저
3   function makeSandwich(){
4        var magicIngredient = "땅콩버터";
5
6        function make(filling){
7           return magicIngredient + " and" + filling;
8        }
9
10       return make("젤리");
11  }
12
13  makeSandwich(); //땅콩버터
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

02. 클로저(Closure)

> 부모함수 수행 종료 후에도
리턴된 내부함수에선
부모함수 변수 접근 및 갱신가능
(함수 리턴, 나중 실행)

예제

```
function makeSandwich(){
    var magicIngredient = "땅콩버터";

    function make(filling){
        return magicIngredient + " and" + filling;
    }

    return make;
}

var func = makeSandwich();
alert(func('바나나')); //땅콩버터 and 바나나
alert(func('젤리')); //땅콩버터 and 젤리
alert(func('마쉬멜로우')); //땅콩버터 and 마쉬멜로우
```

02. 클로저(Closure)

> 클로저란,

함수 자신이 포함하는
스코프의 변수를 추적하는 함수

변수를 저장하는 스코프

예제

```
1
2   //클로저란 함수 자신이 포함하는
3   //스코프의 변수를 추적하는 함수
4   function makeSandwich(magicIngredient){
5       function make(filling){
6           return magicIngredient + " and" + filling;
7       }
8       return make;
9   }
10
11  var hamAnd = makeSandwich("햄");
12  hamAnd("치즈"); //햄 and 치즈
13  hamAnd("머스타드");//햄 and 머스타드
14
15  var chikenAnd = makeSandwich("치킨");
16  chikenAnd("양파"); //치킨 and 양파
17  chikenAnd("피클");//치킨 and 피클
18
19  /////////다음으로 대체가능///////////
20  function makeSandwich(magicIngredient){
21      return function(filling){
22          return magicIngredient + " and" + filling;
23      }
24  }
25
26
27
28
29
30
31
32
```

02. 클로저(Closure)

> **정보은닉, 외부 변수를 private변수처럼** 사용 가능

> 클로저 간에
외부변수 공유 가능(참조저장)

```
1
2   //정보은닉, 외부 변수를 private변수처럼 사용 가능
3   //클로저간에 외부변수 공유 가능(참조저장)
4   function box(){
5         var val = undefined;
6
7         return function(filling){
8               set = function(newval) { val = newval; },
9               get = function() { return val; },
10              type = function() { return typeof va;l }
11        }
12  }
13
14  var b = box();
15  b.type();
16  b.set(98.6);
17  b.get();
18  b.type(); //number
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

03. 변수 호이스팅
(Hosting: 끌어올리다)

> 암묵적으로 블록 내
정의된 변수 선언을
함수 맨 윗 부분으로 끌어올림(Hoisting)

예제

```
function f(){
    var a = 10;

    if(1){
        var a = 12;
        alert(a); // ?
    }

    alert(a); // ?
}
```

03. 변수 호이스팅
(Hosting: 끌어올리다)

> 블록단위 스코프 미지원
자신을 포함하는 함수 스코프에 의존

> 변수의 재선언은 하나의 변수로 처리됨

> 혼란을 막기 위해 변수 선언을
직접 호이스팅

예제

```
function f(){
    for(var i =0, n = 10; i < n; i++){}
    for(var i =0, n = 11; i < n; i++){}
    for(var i =0, n = 12; i < n; i++){}
}

//////////////////////////////////////////

function f(){
    var i, n;
    for(i =0, n = 10; i < n; i++){}
    for(i =0, n = 11; i < n; i++){}
    for(i =0, n = 12; i < n; i++){}
}

//혼란을 막기위해 변수 선언을 직접 호이스팅 권고
```

04. 클로저 내 즉시실행함수

> 각 클로저는
부모 변수의 값이 아닌 참조 저장

예제

```
function wrapElements(a){
     var result = [], i, n;
          for( i = 0, n = a.length; i < n; i++){
               result[i] = function(){
               return a[i];
          }
     }
}

var wrapped = wrapElements([10, 20, 30, 40, 50]);
var f = wrapped[0];
alert(f()); //?
```

04. 클로저 내 즉시실행함수

> 지역 스코프를 만들기 위해 IIFE사용 (블록 스코프 지원을 위한 필수적인 차선책)

예제

```
function wrapElements(a){
     var result = [], i, n;
     for( i = 0, n = a.length; i < n; i++){
          (function(j){
          result[0] = function() { return a[j]; }
          })(i)
     }
      return result;
}

var wrapped = wrapElements([10, 20, 30, 40, 50]);
var f = wrapped[0];
alert(f()); //?
```

Fin