INTERPARK

2017 NOV.

{ .JS }

JavaScript Lab

function { 함수 }

JavaScript Lab 3

01. 함수, 메소드, 생성자

> 함수: 일반적인 함수

> 생성자: new 연산자로 객체 초기화

> 메소드:
함수로 동작하는 객체의 프로퍼티
(메소드 내에서 객체의 프러퍼티 접근 시 this 사용)

예제

```
1
2   //1. 함수
3   function hello(name){
4           return "hello" + name;
5   }
6
7   //2. 생성자
8   var obj = {
9       username: "oleh",
10      hello: function(){
11          return "hello" + this.username;
12      }
13  }
14
15  //3. 메소드
16  function emp(empno, name){
17      this.empno = empno;
18      this.name = name;
19  }
20
21  var u = new emp('N14401','이은학');
22
23
24
25
26
27
28
29
30
31
32
```

02. 고차함수
(High-order-function)

> 함수:
다른 함수를 인자로 받거나
(콜백함수) 결과로 함수를 반환하는 함수
ex) array.sort

> 이미 존재하는
라이브러리의 고차함수의 사용에
익숙해져야함

예제

```
1
2   //2 고차함수
3   function compareNumbers(x, y){
4       if(x < y){
5         return -1;
6       }
7       if(x > y){
8         return 1;
9       }
10      return 0;
11  }
12
13  [3, 1, 4, 1, 5, 9].sort(compareNumbers);
14  //[1,1,3,4,5,9]
15
16  var names = ["eunhak","gusik","nbook"];
17  var upper = [];
18
19  for(var i = 0, n = name.length; i < n; i++){
20    upper[i] = names[i].toUpperCase();
21  }
22
23  upper; //["EUNHAK","GUSIK","NBOOK"];
24
25  ////////////////////////////////////////
26  var names = ["eunhak","gusik","nbook"];
27  var upper = names.map(function(name){
28    return name.toUpperCase();
29  })
30
31  upper; //["EUNHAK","GUSIK","NBOOK"];
32
```

02. 고차함수
(High-order-function)

> 코드에 공통 패턴을
찾고 고차 함수로 대체하는 습관 필요

> 이로 인해
간결한 코드, 높은 생산성 및 가독성 증가

```
1
2   //1)
3   var aIndex = "a".charCodeAt(0);
4   var alphabet = "";
5   for(var i = 0; i < 26; i++){
6     alphabet += String.fromCharCode(aIndex + 1);
7   }
8   //alphabet; //"abcdefghijklmnopqrstuvwxyz"
9
10  //2)
11  var digits = "";
12  for(var i = 0; i < 8; i++){
13    random += String.fromCharCode(
14    Math.floor(Math.random() * 26) + aIndex);
15  }
16
17  ////////////////////////////////////////////////
18
19  function buildString(n, callBack){
20       var result;
21       for(var i = 0; i < n; i++){
22         result += callBack(i);
23       }
24       return result;
25  }
26
27  var alphabet = buildString(26, function(i){
28    return String.fromCharCode(aIndex + 1);
29  });
30
31
32
```

03. Call 메소드

> 특정 객체 메소드를 사용하여 (빌려) 다른 객체의 프로퍼티를 제어할 때 사용

예제

```
var dict = {
  name: "oleh"
};

dict.hasOwnProperty = 1;
//dict.hasOwnProperty("name");
// => dict.hasOwnProperty is not function
var otherDic = {};
otherDic.hasOwnProperty.call(dict, "name"); //true

```

> 위 상황에서 call 함수 미 사용시 특정 객체의 프로퍼티가 변경됨

> 고차함수에서 콜백함수를 위한
다른 객체를 수신 시 call 메소드를 사용하여 제어

예제

```
var table = {
  entries = [],
  addEntry: function(k, v){
    this.entries.push({ key: k, value: v});
  },
  forEach: function(f, thisArg){
    for(var i = 0, n = entries.length; i < n; i++){
      var entry = entries[i];
      f.call(thisArg, entry.key, entry.value);
    }
  }
}

//table1의 entries배열을 table2 entries로 복사
table1.forEach(table2.addEntry, table2);

```

04. Apply 메소드

> 가변인자 함수 호출 시
인자값인 배열을 개별 인자로 불리해주는 역할

> apply 첫번째 인자로
this에 바인딩 될 객체 전달

예제

```
1
2
3
4
5
6
7   ///가변인자 함수///
8   average(1,2,3);
9   average(1,2,3,4,5);
10  average(1,2,3,4,5,6);
11
12  ///가변인자 함수///
13  average([1,2,3]);
14  average([1,2,3,4,5]);
15  average([1,2,3,4,5,6]);
16
17  var scores = getAllScores();
18
19  average.apply(null, scores);
20  //=>average(scores[0],scores[1],scores[2])
21
22
23
24
25
26
27
28
29
30
31
32
```

05. Arguments 객체
(가변인자 함수 생성)

> 함수 호출 시
자바스크립트가 모든 함수에
암묵적인 arguments전달

> 실제 인자가 인덱싱 된 배열

예제

```
function average(){
  for(
      var i = 0, sum = 0,
      n = arguments.length;
      i < n;
      i++
   ){
     sum += arguments[i];
   }
   return sum / n;
}

//////////////////BEST PRACTICE//////////////////
function average(){
  return averageOfArray(arguments);
}
```

06. Bind 함수

> 콜백 함수에 this구문을 참조하는 메소드 전달 시 전달 객체의 스코프 접근 불가

예제

```
var buffer = {
  entries: [],
  add: function(s){
    this.entries.push(s);
  },
  concat: function(){
    return this.entries.join('');
  }
}

var source = ["010","-","9160","-","4423"];
source.forEach(buffer.add) // => ??
buffer.concat();
```

06. Bind 함수

> bind사용으로
특정 객체의 스코프 전달 가능

> bind사용 시
수신자 객체로 바인딩 된
새로운 함수 생성

예제

```
1
2
3   //bind사용으로 특정 객체의 스코프 전달가능
4   //1) 첫번째 방법
5   var source = ["010","-","9160","-","4423"];
6   source.forEach(buffer.add, buffer);
7   buffer.join(); //010-9160-4423
8
9   //2) 두번째 방법(wrapper function)
10  var source = ["010","-","9160","-","4423"];
11  source.forEach(function(s){
12    buffer.add(s);
13  });
14  buffer.join(); //010-9160-4423
15
16  //3) 세번째 방법
17  var source = ["010","-","9160","-","4423"];
18  source.forEach(buffer.add.bind(buffer));
19  buffer.join(); //010-9160-4423
20
21
22
23
24
25  //bind사용시 수신자 객체로 바인딩된 새로운 함수생성
26  buffer.add === buffer.add.bind(buffer); // false
27
28
29
30
31
32
```

06. Bind 함수

> 인자 값이 고정인
함수 호출 시 유용

예제

```
1
2
3
4
5
6
7
8
9
10 //인자값이 고정인 함수 호출 시 유용
11 function simpleURL(protocol, domain, path){
12   return protocol + "://" + domain + path;
13 }
14
15 var url = paths.map(function(path){
16   return simpleURL("http", siteDomain, path)
17 })
18
19 var urls = paths.map(smpleURL.bind(null, "http",
20 siteDomain));
21
22
23
24
25
26
27
28
29
30
31
32
```

Fin