[11/06] JS Language

- 이중 반복문
  - a=1,2,3,4,5 / b =1,2,3 일 경우 각각을 곱하는 결과값 출력

```
for (a=1; a<=5; ++a){
    for (b=1; b<=3; ++b){
        console.log("a*b="+a*b)
    }
}
```

```
a*b=1  a*b=3  a*b=5
a*b=2  a*b=6  a*b=10
a*b=3  a*b=9  a*b=15
a*b=2  a*b=4
a*b=4  a*b=8
a*b=6  a*b=12
```

    - 첫째 for 문은 a=1 로 시작할때 둘째 for 문은 b=1,2,3 이 완료되는 순서
    - a=1 b=1,2,3 / a=2 b=1,2,3 / … / a=5 b=1,2,3 #구구단
    - + : 문자와 문자, 문자와 숫자등의 결합을 의미

  - a=1,2,3,4,5 / b =1,2,3 일 경우 각각을 곱하는 결과값의 합

```
let i=0;
for (a=1; a<=5; ++a){
    for (b=1; b<=3; ++b)
    i+=a*b
}
console.log(i);          90
```

    - for 문안으로 let i=0; 를 선언하면 매번 i=0으로 선언(리셋)되므로 값이 다름

- 반복문 = 배열
  - 배열(array)[1] : 같은 타입의 변수들로 이루어진 유한 집합, 2차원[2]
  - 요소(element) : 배열을 구성하는 각각의 값
  - 앞선 이중 반복문의 a=1,2,3,4,5 / b =1,2,3 의 경우 a=Row(행), b=Col(열)
  - DB의 Table 도 Row 부터 입력됨

1	2	3	4	5
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3

- array & index : [ ]

```
let a=[1,2,3,4];
for(let i=0; i<a.length; i++)
console.log(a[i]);
```

```
1
2
3
4
```

  - a.length : a 의 총길이
  - a[i] : a의 i 번째 값, a[0]=1, a[1]=2, a[2]=3, a[3]=4
  - a=[“one”] : 문자열의 경우 “ ” 이 필요
    - JSON 데이터는 전부 “ ” 씌워서 보냄

---

[1] Array(배열) vs List(리스트) 의 차이 : 간략히 Array는 메모리상에 연속적 저장, List는 비연속적 저장

[2] ex) 1차원(1반만 존재), 2차원(1-1반~1-10반), 3차원(학년), 4차원(학교)

- 배열의 시작 : 0

```
let a=["one", "two", "three", "four"];
for(let i=0; i<a.length; i++){
    console.log(a.length+"/" +i);
    console.log(a[i]);
}
```

```
4/0
one
4/1
two
4/2
three
4/3
four
```

  - i<=a.length 의 경우 에러, 숫자는 반드시 0부터 시작함

```
let a=["one", "two", "three", "four"];
for(let i=0; i<a.length; i++){
    console.log(a.length+"/" +i);
    console.log(a[i]);
    console.log(typeof a[i]);
    console.log(typeof a);
}
```

```
4/0
one
string
object
4/1
two
string
object
4/2
three
string
object
4/3
four
string
object
```

  - array=객체(object), object 도 하나의 결합체 one == o+n+e
  - 배열(array)의 주소는 0부터 시작 ex) [0,1,2,3] / 해당 배열의 갯수는 4개

- 배열 & 출력형식

```
let a=[3];
a[5] = 456;
console.log(a.length);
```

```
6
```

```
let a=[3];
a[5] = 456;
for (i=0; i<a.length; i++){
    console.log("a[%d] : %s", i ,a[i]);
}
```

```
a[0] : 3
a[1] : undefined
a[2] : undefined
a[3] : undefined
a[4] : undefined
a[5] : 456
```

  - a=[3] : 0번째 자료값으로 3으로 선언, 즉 a[0]=3
  - a[5] =456 : 5번째 자료값으로 456 선언
    - 출력시 a[1],a[2],a[3],a[4] 들을 선언하지 않아도 자동 undefined[3] 지정
  - 출력형식 : [%d] : integer / [%s] : string / [%f] : float / [%c] : char
    - a[%d] : [%s] , i , a[i] : [%d] 는 i 와 연결, [%s] 는 a[i] 와 연결
    - 출력할 형식이 integer 이므로 %d 를 작성, i 값이 출력
    - 출력할 형식이 string 이므로 %s 를 작성, a[i] 값이 출력
  - 빈 배열 : a=[ ] 일 경우 length 는 0, 빈 배열만 만든 상태

- 구조 분해 할당 : 배열이나 객체의 속성을 해체하여 그 값을 개별 변수에 담을 수 있게 하는 JavaScript 표현식

```
let a = [3,4,5];
[a1, a2] = a;
console.log("%d %d" , a1, a2);
```

```
3 4
```

  - 묶여진 부분중 일부만 출력
  - let [a1, a2, a3] = a; 의 경우 a 가 정의되지 않은 상태

---

[3] undefined : 공간 미지정, 값 미지정 null : 공간 지정, 값 미지정

- …a3 : a3 ~ 나머지 전부를 포함

```
let a = [1,2,3,4,5,6,7];
let [a1, a2, ... a3] = a;
console.log("%d %d", a1, a2);
console.log(a3);
```

```
1 2
[ 3, 4, 5, 6, 7 ]
```

- 배열의 법칙 : 좌측 우선 (고정), 후선언 채택 (덮어쓰기)
  - let a,b;
    a = 5;
    b = 6;
    let temp = 5;
    a = 6;
    b = 5;
    console.log(6,5);

```
let a, b;
a = 5;
b = 6;
let temp = a;
a = b;
b = temp;
console.log(a,b);
```

```
6 5
```

- …a

```
let i = 3, j =4;
let a = [5,6,7];

let a1 = [i,j,a];
console.log(a1);

let a2 = [i,j,...a];
console.log(a2);
```

```
[ 3, 4, [ 5, 6, 7 ] ]
[ 3, 4, 5, 6, 7 ]
```

  - a2 : array 내 호출시 [ ] 발생하지 않고 value 만 호출

- array의 length

```
let i = 3, j =4;
let a = [5,6,7];

let a1 = [i,j,a];

console.log(a1);
console.log(a1.length);

let a2 = [i,j,...a];
console.log(a2);
console.log(a2.length);
```

```
[ 3, 4, [ 5, 6, 7 ] ]
3
[ 3, 4, 5, 6, 7 ]
5
```

  - let a1 = [i,j,a] 의 length 내포된 것(a=5,6,7)을 포함하지 않고 갯수 그대로 반영(결과값 3)

- toString() : 배열을 제외하고 출력([ ]을 제거)

```
let a1 = [1,2,3];
let a2 = ["a","b","c"];
console.log(a1);
console.log(a1.toString());
console.log(a2.toString());
```

```
[ 1, 2, 3 ]
1,2,3
a,b,c
```

- splice : 쪼개기

```
let a = [0,1,2,3];
a.splice(1,0,"a")
console.log(a);
console.log(a.toString());
```

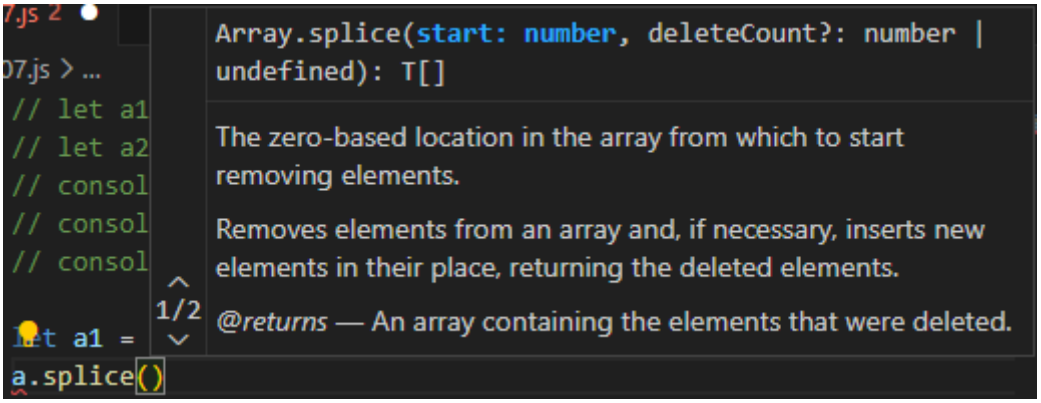
```
[ 0, 'a', 1, 2, 3 ]
0,a,1,2,3
```

  - 주소 : a 의 1번째 값 / 삭제 : 0이라 삭제하지 않음 / 삽입 : "a"

```
let a = [0,1,2,3];
a.splice(1,1,"a")
console.log(a);
console.log(a.toString());
```

```
[ 0, 'a', 2, 3 ]
0,a,2,3
```

  - 주소 : a 의 1번째 값 / 삭제: 1이라 삭제 / 삽입 : "a"



  - splice시 나타나는 화면

- slice : 반복

```
let b = [0,1,2,3];
console.log(b.slice(0,1));
console.log(b.slice(0,2));
console.log(b.slice(1,2));
console.log(b.slice(1,3));
```

```
[ 0 ]
[ 0, 1 ]
[ 1 ]
[ 1, 2 ]
```

  - (시작값, 종료값)
  - (0, 1) : 0 에서부터 1번만 : 0
  - (1, 3) : 1 에서부터 3번만 : 1 ,2 // 앞선 0은 시작값이 1 이므로 미포함
  - 배열 복제 : let b = a.slice(0); 0번부터 끝나는 지점을 미설정시 사실상 "복사"

- push : 빈 배열에 항목을 추가

```
let a=[];
for (let i=0; i<10; i++){
    a.push(i);
}
console.log(a);
console.log(a.toString());
```

```
[
  0, 1, 2, 3, 4,
  5, 6, 7, 8, 9
]
0,1,2,3,4,5,6,7,8,9
```

  - 현재 a=[ ] 으로 빈 값, 출력시 [0,1,2,3,4,5,6,7,8,9]
  - 만약 a=[10] 을 지정시 [10,0,1,2,3,4,5,6,7,8,9] 로 출력

- pop : 차감식

```
let a=[];
for (let i=0; i<10; i++){
    a.push(i);
}

for (let j=a.length; j>0; j--){
    a.pop(j);
    console.log(j);
}
```

```
10
9
8
7
6
5
4
3
2
1
```

```
let a=[0,1,2,3,4,5,6,7,8,9];

for (let j=9; j>0; j--){
    a.pop(j);
    console.log(a);

}
```

```
[
  0, 1, 2, 3, 4,
  5, 6, 7, 8
]
[
  0, 1, 2, 3,
  4, 5, 6, 7
]
```

```
[
  0, 1, 2, 3,
  4, 5, 6
]
[ 0, 1, 2, 3, 4, 5 ]
[ 0, 1, 2, 3, 4 ]
[ 0, 1, 2, 3 ]
[ 0, 1, 2 ]
```

```
[ 0, 1 ]
[ 0 ]
```

  - j - - : 증감식(i++)과 반대로 하나씩 줄임

```
let a=[10];
for (let i=0; i<10; i++){
    a.push(i);
}
// console.log(a);
// console.log(a.toString());

for (let j=a.length; j>0; j--){
    a.pop(j);
    console.log("length %d array %s", j, a);
}
```

```
length 11 array [
  10, 0, 1, 2, 3,
  4, 5, 6, 7, 8
]
length 10 array [
  10, 0, 1, 2, 3,
  4, 5, 6, 7
]
length 9 array [
  10, 0, 1, 2,
  3, 4, 5, 6
]
```

```
length 8 array [
  10, 0, 1, 2,
  3, 4, 5
]
length 7 array [ 10, 0, 1, 2, 3, 4 ]
length 6 array [ 10, 0, 1, 2, 3 ]
length 5 array [ 10, 0, 1, 2 ]
length 4 array [ 10, 0, 1 ]
length 3 array [ 10, 0 ]
length 2 array [ 10 ]
length 1 array []
```

  - length 와 array 는 보기 좋게 쓴 글자에 불과

- concat : 배열과 배열을 하나의 배열로

```
let a = [1,2,3];
let b = [4,5,6];
let c = a+b;
let d = a.concat(b);
console.log(a);
console.log(b);
console.log(c);
console.log(d);
```

```
[ 1, 2, 3 ]
[ 4, 5, 6 ]
1,2,34,5,6
[ 1, 2, 3, 4, 5, 6 ]
```

- sort

```
let a = [31, 11, 25, 7, 1, 2, 3, 13, 9];
a.sort();
console.log(a);
```

```
1, 11, 13, 2, 25,
3, 31,  7, 9
```

  - string 처리 순서는 2진수 기준의 배열
  - 자릿수가 다를 경우 주로 발생, 01,02,03 처럼 자릿수 통일이 필요
  - 문자와 숫자의 결합된 경우 sort(정렬)시 주로 발생
  - 개발 단계부터 자릿수를 통일하거나 function을 이용한 작업이 필요