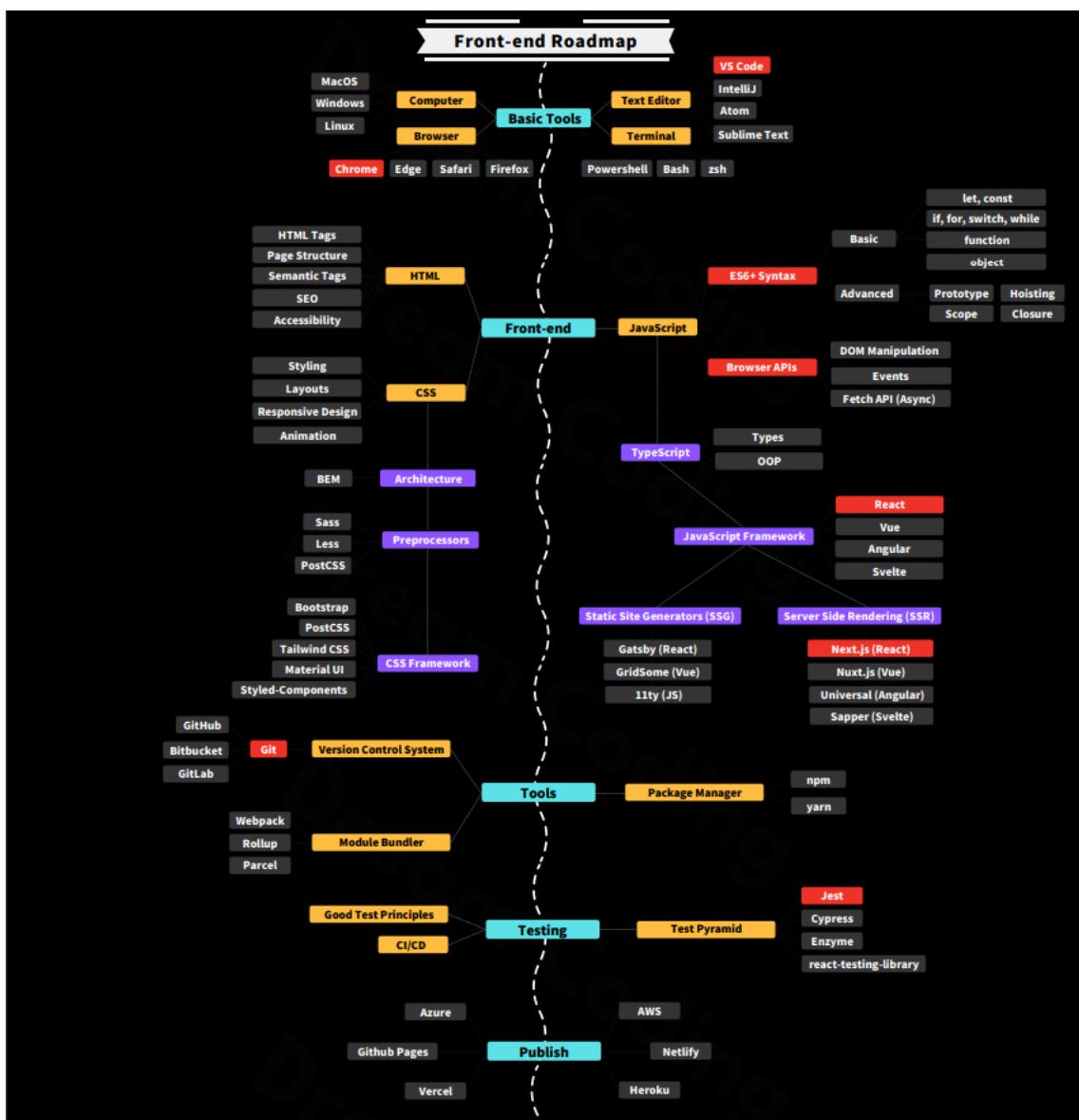




JavaScript / JQuery

① 작성일시	@2023년 4월 10일 오후 7:13
▽ 강의 번호	SBS아카데미 컴퓨터 학원
▽ 유형	강의
📎 자료	
☒ 복습	<input type="checkbox"/>

[Front-end RoadMap]



[웹 컴파인 #3 : JavaScript]

- 자바스크립트의 특징

- ⇒ 모든 웹 브라우저에서 작동
- ⇒ 웹 개발 뿐 아니라 다양한 용도의 프로그램을 만들 수 있다
- ⇒ 다양한 자바스크립트 공개 API 사용 가능
- ⇒ 다양한 라이브러리와 프레임워크 사용 가능

• 자바스크립트의 입/출력

- ⇒ `prompt()` 함수 : 사용자에게 값을 입력 받을 때 가장 쉽게 사용할 수 있는 함수
- ⇒ `alert()` 함수 : 웹 브라우저 화면에서 간단한 알림 내용 출력
- ⇒ `document.write()` 함수 : 결과값을 웹 브라우저 화면에 출력
- ⇒ `console.log()` 함수
 - ⇒ 브라우저 검사도구(f12)에서 확인 가능한 영역
 - ⇒ 콘솔창을 통해서 프로그램의 오류를 보거나 값을 테스트 한다
 - ⇒ ;(세미콜론)은 명령이 끝나는 영역을 알려주는 마침표 역할을 한다
 - ⇒ 자바스크립트는 매우 유한 언어라서 세미콜론 없이 실행 가능
 - ⇒ 명령이 길어지면 구분이 어렵기 때문에 짹는 것을 권장

• 주석의 종류

- 1) // : 행단위 주석
- 2) /* 블럭단위 주석 */

• 변수

1. 변수 ?
 - Variable
 - let, var라는 키워드를 통해서 선언
 - let 변수의 이름 = 변수에 담을 내용
 - 변수의 이름은 id와 같다. 유니크한 값이어야 한다. 중복이 안된다

2. 변수 상세

- 변수에는 어떠한 값이든 넣을 수 있다
- 변수의 이름을 호출해서 사용할 수 있다
- 변수 생성자의 이름과 호출시의 이름이 같아야 한다

3. 키워드의 차이점

- **let, var, const** : 변수가 최초에 선언될 때에 한번만 사용되는 키워드
- **let, var** : 단순 변수 선언시에 사용되는 키워드
 - 이후 변수의 값을 바꿀때에는 '변수이름 = 바꿀 내용'으로 재할당하여 사용 가능하다
- **const** : 상수의 선언에 사용되는 키워드
 - `let, var`와 다르게 고정이 필요한 값을 선언할 때 사용된다
 - 한번 선언하면 재할당 불가

4. 자바스크립트에서만 사용?

- 변수는 모든 프로그래밍 언어에서 사용되는 개념

- 언어별 선언 및 사용 방식의 차이

5. 변수의 선언 방법

- var** 변수이름 = 변수에 저장할 내용
- let** 변수이름 = 변수에 저장할 내용

※ 통상적으로 **var**보다 **let**을 더 많이 사용

※ **var**는 2016년 이전에 주로 사용 (**let**은 **var**의 단점 보완)

1. 변수선언

```
let americano = 4500;      // 초기 선언
let latte = 4500;
let capuccino = 4500;
let cookie = 3000;

americano = 4500;          // 재할당
※재할당 시 let을 사용하면 오류 발생

console.log(americano * 3 + cookie * 2);
```

2. 상수 선언

```
const CUP = 10000;
const cup = 5000;
console.log(CUP + americano);
console.log(cup + americano);
※대소문자 결과 다르게 출력
```

6. 변수와 상수의 차이점

- 변수는 값의 재할당 가능하나 상수는 불가능
- 상수는 초기 선언 시 무조건 값을 지정해주는 것이 좋다

7. 변수와 상수의 이름 규칙

- 시작은 영문 또는 언더바(_) 또는 \$로 시작
- 두번째부터 숫자 가능
- 대소문자 구분
- 고유로 지정된 키워드는 사용 불가능

7-1) 변수와 상수의 이름 관례

- 의미 없는 이름 사용하지 않기
- 너무 추상적인 이름은 좋지 않다
- 모든 변수의 이름은 **카멜케이스(camelCase)** 또는 **스네이크케이스(Snake_case)** 또는 **파스칼(VariableNameValue)**을 사용하기
- 상수의 이름은 대문자만 사용

• 자료형 (Date Type)

=> 변수 안에 저장할 수 있는 다양한 형태의 자료형

• 자료형 타입

1. String 타입

- 문자열, 따옴표안에 들어가있는 데이터

- 따옴표 안에 들어간 것들은 모두 문자열로 인식
- ex) "123" + 1 의 결과 값은 1231이 출력
- 큰따옴표("")와 작은따옴표('')를 구분하지 않는다. 무조건 문자열로 인식

```
console.log(1234);
console.log("1234");
console.log(false);
console.log("1" + 1);
```

※ 문자열의 경우 검은색, 숫자의 경우 파란색으로 출력

1234	자료형.html:76
1234	자료형.html:77
>	

2. Number 타입

- 숫자, 양수와 음수, 소수, 모두다 숫자 타입으로 들어간다
- 123 + 1 의 결과 값은 124가 출력

3. Boolean 타입

- 논리 연산에 쓰이는 데이터
- true(참), false(거짓) 두개의 값으로 표현
- 불대수, 불타입, 블린형 등으로 불러진다
- 상수형 데이터의 가장 대표적인 데이터

4. 기타

- 배열(array) : [] 로 묶어두는 자료형
- 객체(object) : {} 로 묶어두는 자료형

• 연산자(Operator)

```
let number = 10;
console.log(++number);

let a = 111;
let b = "111";
console.log(a == b);      * a는 숫자, b는 문자열이라 false 결과 출력
console.log(a === b);
```

11	03 연산자.html:87
true	03 연산자.html:91
false	03 연산자.html:92

기본 연산자			
순서	연산자	사용의 예	설명
1.	-	: C = -A	A 값이 양수이면 음수로, 음수이면 양수로 변환
2.	-	: C = A - B	A에서 B를 뺀 차를 C에 저장. 뺄셈 연산
3.	+	: C = A + B	A와 B의 합을 C에 저장. 덧셈 연산
4.	*	: C = A * B	A와 B의 곱을 C에 저장. 곱셈 연산
5.	/	: C = A / B	A를 B로 나눈 몫을 C에 저장. 나눗셈 연산
6.	%	: C = A % B	A를 B로 나누었을 때 나머지를 C에 저장. 나머지 연산
7.	++	: C = ++A	A 값에 1을 더한 값을 C에 저장. 전위 증가
8.	--	: C = --A	A 값에 1을 뺀 값을 C에 저장. 전위 감소
9.	++	: C = A++	A 값에 1을 더한 값을 C에 저장. 후위 증가
10.	--	: C = A--	A 값에 1을 뺀 값을 C에 저장. 후위 감소

관계 연산자			
순서	연산자	사용의 예	설명
1.	==	: A == B	A와 B의 값이 같은지 비교 (동등비교연산자)
2.	==	: A === B	A와 B의 값뿐만 아니라 자료형도 같은지 비교 (일치연산자)
3.	!=	: A != B	A와 B의 값이 다른지 비교
4.	!==	: A !== B	A와 B의 값뿐 아니라 자료형도 다른지 비교
4.	>	: A > B	A가 B보다 큰지 비교
5.	>=	: A >= B	A가 B보다 크거나 같은지 비교
6.	<	: A < B	A가 B보다 작은지 비교
7.	<=	: A <= B	A가 B보다 작거나 같은지 비교

논리 연산자			
순서	연산자	사용의 예	설명
1.	&&	: A && B	AND 연산, A와 B 둘 다 참일 때 참을 출력
2.		: A B	OR 연산, A와 B 둘 중에 하나만 참이면 참을 출력
3.	!	: !A	NOT 연산, A가 참이면 거짓, 거짓이면 참을 출력

복합 대입 연산자			
순서	연산자	사용의 예	설명
1.	+=	: A += B	A = A + B와 같다. A를 B만큼 증가
2.	-=	: A -= B	A = A - B와 같다. A를 B만큼 감소
3.	*=	: A *= B	A = A * B와 같다. A를 B만큼 곱해라
4.	/=	: A /= B	A = A / B와 같다. A를 B만큼 나눈 몫을 재할당
5.	%=	: A %= B	A = A % B와 같다. A를 B만큼 나눈 나머지를 재할당

• 호이스팅(Hoisting)

1. 호이스팅 : 자바스크립트의 특이한 특징 ?

- 자바스크립트는 문서가 실행되면서 코드를 실행하기 전에 변수와 함수를 먼저 메모리에 저장해두는 과정(단계)
- 함수가 실행되기 전에 안에 있는 변수들을 범위의 최상단으로 옮기는 개념이 호이스팅이다

2. let과 var의 차이

- 2015년도에 자바스크립트가 ES6 문법으로 업그레이드 되면서 생겨난 것이 let이다
- var의 문제점

```

var a = 1;
console.log(a);
※문제 없는 코드이다

console.log(a);
var a = 1;
console.log(a);
라고 입력되면 undefined라는 코드가 먼저 나오게 된다.
같이 안되는 코드 같지만, 콘솔창에 출력하기 전에 v8엔진이 선언된 변수 a를 먼저 읽어들이기 때문에
오류가 아니라 undefined(찾을 수 없음)라는 문구로 출력 된다.
자바스크립트에서 호이스팅을 할 때에 변수의 선언과 초기화를 같이 시켜버린다.
그리고서 할당은 나중에 실제 코드에서 변수가 선언되어질 때 할당하게 된다.

console.log(a);
a = 1;
var a;
console.log(a);
이것 또한 말도 안되는 코드 같지만, 자바스크립트는 이를 이해한다

결론 1) 자바스크립트는 매우 유한 언어이다. 허용범위가 크다.

```

```
var의 문제점이 더 있다면 지역변수와 전역변수의 개념이 확실하지 않다는 점이다.  
var의 문제점 마지막으로는 중복선언을 허용한다는 점이 있다.
```

※ 전역변수 : 블럭 밖에서 아무것도 없이 그냥 선언되어진 변수, 어디에서나 접근 할 수 있고, 사용할 수 있는 변수
※ 지역변수 : 함수나 기타 등등 블럭 범위의 안에서 선언되어진 변수, 해당되는 지역 안에서만 쓸 수 있는 변수

- 위에서 언급된 var의 문제점들이 엄청나기 때문에 let을 만들게 되었다
- 실제 2015년 이후 var보다 let을 사용해 줄 것을 적극적으로 권장하고 있다
- 논리적이고 상식적으로 작성된 코드라면 var를 let으로 바꾸었을 때 아무런 문제가 없어야 한다
- let은 Temporal Death Zone(TDZ)라는 개념을 만들었다
- 호이스팅이 이루어진 후에 선언되어진 변수라는 것은 알고 있지만, 실제 변수가 선언되어지기 전이라면 오류를 일으킨다
- 호출되기 전이니까 '호출되어지면 사용하겠다'라는 느낌이라고 이해하면 좋다

3. 결론

- var를 쓰지말고 let을 사용

※ 2일차 과제 풀이

```
// 문제 1) 나이 계산기  
const BIRTH_YEAR = 1985;  
let year = 2023;  
let age = year - BIRTH_YEAR + 1;  
  
console.log(age);  
// 한줄 공백  
console.log('')  
  
// 문제 2) 음료 칼로리 계산기  
let espresso = 10;  
let milk = 170;  
let chocolate_syrup = 50;  
let whipped_cream = 60;  
  
※ 값이 변하지 않을 것 같은 변수는 상수로 선언  
const AMERICANO = espresso;  
const LATTE = espresso;  
const MOCCA = espresso + chocolate_syrup + milk;  
const WHIPPED_MOCCA = espresso + chocolate_syrup + milk + whipped_cream;  
  
console.log(AMERICANO);  
console.log(LATTE);  
console.log(MOCCA);  
console.log(WHIPPED_MOCCA);
```

• 배열

1. 배열이란 ?

- 변수에 담을 수 있는 여러가지 데이터 중 하나
- 문자, 숫자, 블린 외 하나의 자료형

2. 배열의 탄생배경

- 변수에는 일반적인 데이터를 하나씩만 담을 수 있게 되어 있다
- 하지만 배열에는 여러가지의 데이터들을 배열이라는 하나의 이름으로 담을 수 있다

3. 선언

```
let fruit = ['banana', 'apple', 'grape', 'mango'];
```

- 배열도 숫자나 문자처럼 일종의 자료형이므로 변수에 할당한다

```

<script>
    let fruit1 = 'banana';
    let fruit2 = 'apple';
    let fruit3 = 'grape';
    let fruit4 = 'mango';

    let fruit = ['banana', 'apple', 'grape', 'mango'];

    console.log(fruit);
    console.log(fruit[0]); // 배열값의 참조

    fruit[1] = 'cherry' // 배열값의 수정
    console.log(fruit[3]);
</script>

```

▼ (4) ['banana', 'apple', 'grape', 'mango'] ⓘ
0: "banana"
1: "cherry"
2: "grape"
3: "mango"
length: 4
▶ [[Prototype]]: Array(0)

banana	05 배열.html:36
mango	05 배열.html:39

Ex01. 주어진 단어에 특정 알파벳 갯수 카운트 프로그램

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ex04. 주어진 단어에 특정 알파벳 갯수 카운트 프로그램</title>
</head>
<body>

<script>
    // 주어진 단어에 알파벳 'E', 'X'가 몇번 들어간지 카운트하는 함수
    function countCharacter(word, findStr){
        // 결과값을 담을 변수
        let count = 0;
        // console.log(word.length)

        // 주어진 단어의 0번째 값부터 찾는 값과 일치여부 순차적 비교
        for(let i = 0; i < word.length; i++){
            // 주어진 단어를 대문자로 변경 후 0번째부터 순차적 비교
            // a      E
            if(word[i].toUpperCase() === findStr.toUpperCase()){
                // 주어진 단어의 0번째부터 찾는 값과 순차적으로 비교하여 일치하면 카운트 증가
                count++;
            }
        }
        return count;
    }

    // 주어진 단어에 알파벳 'A'가 몇개 들어간지 카운트하는 함수
    function countA(finStr){
        // countA 함수가 실행되면 countCharacter 함수를 호출하는 것으로 중복되는 기능을 나열하는 것을 최소화
        return countCharacter(finStr, 'A');
    }

    /*
        let count = 0;
        for(let i = 0; i < finStr.length; i++){
            if(finStr[i].toUpperCase() === 'A'){
                count++;
            }
        }
        return count; */
    }

    console.log(countCharacter('AbaCedEA', 'E')); // 들어간 횟수 카운트
    console.log(countCharacter('AbaCedEA', 'X'));
    console.log(countA('AbaCedEA')) // 단어 A가 몇번 들어갔는지?

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

```

Ex02. 주어진 배열에서 특정조건에 맞는 값을 찾아 출력하는 프로그램

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ex05. 주어진 배열에서 특정조건에 맞는 값을 찾아 출력하는 프로그램</title>
</head>

```

```

<body>

<script>
/*
기본 Logic
1. 주어진 11개의 문자열(생년월일 8자리, 이름 제공)
2. 이중에서 미성년자만 골라서 출력
- 상세 조건 : 2022년 기준으로 03년생까지 성인, 04년생부터 미성년자
*/


var birthdayDate = [
    '20050309자수', '19960828제니', '20061112로제', '19960209리사', '20060719안유진', '19981216가을',
    '19971201레이', '20061204장원영', '20061204리즈', '19951218이서', '20080803손예진'
]

function printGirl(arr){
    console.log("미성년자 명단")
    for(let i = 0; i < arr.length ; i++){
        // 주어진 배열에서 0번째 ~ 3번째까지 총 4자리(즉, 태어난 년도값 반환)
        // console.log(arr[i].substr(0,4))

        // 주어진 문자열 값을 조건년도(숫자값)와 비교하기 위한 형변환 필요
        if(Number(arr[i].substr(0, 4) > 2005)){
            console.log(arr[i]);
        }
    }
}

printGirl(birthdayDate);

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

```

※ 3일차 과제 풀이

```

let animals= ["Aardvark", "Albatross", "Alligator", "Alpaca", "Ant", "Ape", "Armadillo", "Donkey", "Baboon", "Badger", "Barracuda", "Bat", "Bear",
    "Clam", "Cobra", "Cockroach", "Cod", "Cormorant", "Dugong", "Dunlin", "Eagle", "Echidna", "Eel", "Eland", "Elephant", "Elk", "Emu",
    "Heron", "Herring", "Hippopotamus", "Hornet", "Horse", "Kangaroo", "Kingfisher", "Koala", "Kookabura", "Moose", "Narwhal", "Newt",
    "Red deer", "Sandpiper", "Sardine", "Sparrow", "Spider", "Spoonbill", "Squid", "Squirrel", "Starling", "Stingray", "Tiger", "Toad"

// 문제 1) 어레이에 마지막 아이템 "Zebra" 제거하기
animals.pop();
console.log(animals);

// 문제 2) 주어진 어레이에 "Dog" 추가하기
animals.push('Dog');
console.log(animals);

// 문제 3) 주어진 어레이에 "Mosquito", "Mouse", "Mule" 추가하기
animals.push('Mosquito', 'Mouse', 'Mule' );
console.log(animals);

// 문제 4) 해당 어레이에는 "Human"이 있는가?
console.log(animals.includes('Human'));

// 문제 5) 해당 어레이에는 "Cat" 이 있는가?
console.log(animals.includes('Cat'));

// 문제 6) "Red deer"을 "Deer"로 바꾸시오
// console.log(animals[77]);
console.log(animals.indexOf('Red deer')) // 'Red deer' 인덱스 위치(값) 확인
animals[animals.indexOf('Red deer')] = "Deer"
// animals[77] = 'Deer'
console.log(animals[77]);

// 문제 7) "Spider"부터 3개의 아이템을 기준 어레이에서 제거하시오
// console.log(animals[81], animals[82], animals[83], animals[84]);
animals.splice(animals.indexOf("Spider"), 3);
// animals.splice(81, 3);
console.log(animals);

// 문제 8) "Tiger"이후의 값을 제거하시오
// console.log(animals[84], animals[85], animals[86], animals[87], animals[88], animals[89], animals[90], animals[91], animals[92], an
animals.splice(animals.indexOf("Spider"));
console.log(animals);
// animals.splice(84);
// console.log(animals[84], animals[85], animals[86], animals[87], animals[88], animals[89], animals[90], animals[91], animals[92], an

// 문제 9) "B"로 시작되는 아이템인 "Baboon"부터 "Bison"까지 가져와 새로운 어레이에 저장하시오
// console.log(animals[8], animals[9], animals[10], animals[11], animals[12], animals[13], animals[14], animals[15]);
// let new_animals = [animals[8], animals[9], animals[10], animals[11], animals[12], animals[13], animals[14], animals[15]];

```

```
// console.log(new_animals);
let newList = animals.slice(animals.indexOf("Baboon"), animals.indexOf("Bison") + 1);
// +1을 해주어야 마지막 요소인 'Bison' 까지 포함하는 것
console.log(newList)
```

Ex03. 이메일과 패스워드를 검증하는 자바스크립트 코드

```
/*
기본 Logic
1. email과 password는 '문자열'
2. email에는 반드시 '@'가 포함되어야 함
3. email에 '@'가 없을 경우 '이메일 주소를 다시 확인해주세요' 메세지 팝업
4. 비밀번호 길이 체크
5. password에는 8~12자리가 들어가야 함
6. 8~12 자리가 아니라면 '비밀번호는 8~12 자리여야 합니다' 메세지 팝업
*/

function validate(email, password){
    // 입력받은 이메일과 패스워드를 저장할 배열 생성
    const message = [];

    // 입력된 이메일 정보에 '@' 포함 여부 확인 후 조건에 해당되는 메세지를 배열의 마지막에 추가
    if(email.indexOf "@" == -1){
        message.push("이메일 주소를 다시 확인해주세요.");
    }else {
        message.push("이메일 주소를 올바르게 입력했습니다.");
    }

    // 입력된 패스워드 길이가 8 ~ 12자리 여부 확인 후 조건에 해당되는 메세지를 배열의 마지막에 추가
    if(password.length < 8 || password.length > 12){
        message.push("비밀번호는 8 ~ 12자리여야 합니다.");
    }else {
        message.push("비밀번호를 올바르게 입력했습니다.");
    }

    // 각 조건에 부합하는 메세지 결과 출력
    console.log(message)
    return message;
}

validate("sjqcl3310@naver.com", "1q2w3e4r5t");
// 결과 #1 : [ '이메일 주소를 올바르게 입력했습니다.', '비밀번호를 올바르게 입력했습니다.' ]

validate("sjqcl3310.com", "122333");
// 결과 #2 : [ '이메일 주소를 다시 확인해주세요.', '비밀번호는 8 ~ 12자리여야 합니다.' ]

validate("sjqcl3310@daum.net", "12233323423ed3423423");
// 결과 #3 : [ '이메일 주소를 올바르게 입력했습니다.', '비밀번호는 8 ~ 12자리여야 합니다.' ]

validate("sjqcl3310.com", "abdd");
// 결과 #4 : [ '이메일 주소를 다시 확인해주세요.', '비밀번호는 8 ~ 12자리여야 합니다.' ]
```

• 객체(Object)

※ 객체 : 데이터 타입 중 하나 각각의 기능들의 집합(key, value로 구성)

1. 객체가 필요한 이유 ?

- 하나의 데이터에 많은 정보가 필요한 경우 객체라는 것으로 해결이 가능하다
- 사실 하나의 정보로 이루어진 데이터는 없다
- '나'라는 사람에 대해 표현하기 위해서도 이름, 나이, 사는 곳 등등 많은 정보가 필요하다
- 객체는 관련있는 정보들을 묶어서 하나의 데이터로 표현하기 위해서 탄생하게 되었다
- 자바스크립트는 모든 것이 객체다
- Window 객체 = 전역객체(Global Object)

• DOM(Document Object Model)

⇒ 진입점 역할을 한다

⇒ html 요소를 객체화 한것

※ Window와 DOM 객체 비교

- Window의 경우 내부에 수많은 인스턴스 존재 (필요에 따라 호출 후 사용)
- DOM의 경우 html 태그로 구성

```
> console.log(window);  
VM140:1  
Window {window: Window, self: Window, document: document, name: '', location: Location, ...} ⓘ  
▶ $: f (e,t)  
▶ alert: f alert()  
▶ atob: f atob()  
▶ blur: f blur()  
▶ btoa: f btoa()  
▶ caches: CacheStorage {}  
▶ cancelAnimationFrame: f cancelAnimationFrame()  
▶ cancelIdleCallback: f cancelIdleCallback()  
▶ captureEvents: f captureEvents()  
▶ chrome: {loadTimes: f, csi: f}  
▶ clearInterval: f clearInterval()  
▶ clearTimeout: f clearTimeout()
```

※ DOM을 객체 형태로 보는 방법

```
> console.log(document);  
VM208:1  
▼ #document  
<!DOCTYPE html>  
<html lang="ko">  
▼ <head>  
  <meta charset="UTF-8">  
  <meta http-equiv="X-UA-Compatible" content="IE=edge">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Ex07. LaunchPad</title>  
  <link rel="stylesheet" href="style.css">  
</head>  
▼ <body>
```

```
> console.dir(document);
```

VM414:1

```
▼ #document ⓘ
  ► jQuery370062287962562855631:
  ► location: Location {ancestor
    URL: "http://127.0.0.1:5501/
  ► activeElement: body
  ► adoptedStyleSheets: Proxy(Ar
    alinkColor: ""
  ► all: HTMLAllCollection(51) [
  ► anchors: HTMLCollection []
  ► applets: HTMLCollection []
  baseURI: "http://127.0.0.1:5
  bgColor: ""
  ► body: body
    characterSet: "UTF-8"
    charset: "UTF-8"
    childElementCount: 1
  ► childNodes: NodeList(2) [<!D
  ► children: HTMLCollection [ht
  compatMode: "CSS1Compat"
  contentType: "text/html"
  cookie: ""
```

- **dir** 메소드는 문자열 형식으로 표시

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. On the left, the 'Elements' panel shows the DOM tree. On the right, the 'Console' panel displays the output of a script.

The script in the 'Sources' tab (index2.html) contains the following code:

```
36 console.log(arr);
37 console.dir(arr);
38 console.log('----- obj -----');
39 console.log(obj);
40 console.dir(obj);
41 console.log('----- func -----');
42 console.log(func);
43 console.dir(func);
```

The output in the 'Console' panel shows:

- Line 36: `console.log(arr);`
- Line 37: `console.dir(arr);`
- Line 38: `----- obj -----`
- Line 39: `obj` (with expanded object properties: name: 'Hello', email: 'hello@naver.com')
- Line 40: `----- func -----`
- Line 42: `func` (with expanded function properties: arguments: null, caller: null, length: 0, name: "func", prototype: {constructor: f}, [[FunctionLocation]]: index2.html?_ijt=5tp_d=RELOAD_ON_SAVE:43, [[Prototype]]: f (), [[Scopes]]: Scopes[2])
- Line 43: `func` (with expanded function properties: arguments: null, caller: null, length: 0, name: "func", prototype: {constructor: f}, [[FunctionLocation]]: index2.html?_ijt=5tp_d=RELOAD_ON_SAVE:43, [[Prototype]]: f (), [[Scopes]]: Scopes[2])

A red arrow points from the 'func' entry in the expanded object to the 'func' entry in the expanded function, indicating they are the same object.

※ **console.log** 와 **console.dir**의 비교

```

f func() { index2.html? ijt=5tp_d=RELOAD_
    }
    console.log('I Love Hello');
}

▼ f func() ⓘ
arguments: null
caller: null
length: 0
name: "func"
► prototype: {constructor: f}
[[FunctionLocation]]: index2.html? ijt=5tp_d=RELOAD_
► [[Prototype]]: f ()
► [[Scopes]]: Scopes[2]

```

- log 메소드의 경우 쉼표로 구분된 모든 정보를 출력해주는것에 비하여 dir 메소드의 경우 가장 처음에 선택한 요소 (**첫번째 전달된 값만 기억**)에 대한 정보만 전달해준다

```

Hello 123 true /* (2) [1, 2, 3]
* {name: 'Hello', email: 'hello@naver.com'} f f
    console.log('I Love Hello');

Hello
> 

```

File Tree:

- baseModePoke
- css
- cssStyle
- ED1
- EventDay
- Introduce
- Login
- LunchPad
 - audio
 - css
 - images
 - index.html
 - selecter
 - test
 - travel
 - AlanWalker-TheDrum.mp3
 - index.html
 - index2.html
 - image1.png
 - image2.png
 - project1.png
- External Libraries
- Scratches and Consoles

```

32   console.log('----- bool -----');
33   console.log(bool);
34   console.dir(bool);
35   console.log('----- arr -----');
36   console.log(arr);
37   console.dir(arr);
38   console.log('----- obj -----');
39   console.log(obj);
40   console.dir(obj);
41   console.log('----- func -----');
42   console.log(func);
43   console.dir(func);*/
44
45   console.log(str, num, bool, arr, obj, func);
46   console.dir(str, num, bool, arr, obj, func);

```

- log와 dir 메소드의 가장 큰 차이는 DOM 객체를 표현하는 과정에서 확인할 수 있다
- 값에 조금 더 중점을 둔 log 메소드는 html 태그 형태로 출력되며, 객체의 속성에 좀 더 중점을 둔 dir 메소드는 객체 형태로 출력된다
- 즉, 콘솔에서 값을 확인하고 싶다면 log를 객체 속성을 확인하고 싶을 경우 dir 메소드를 활용하는 것을 추천

```

> console.log(myDOM);
console.dir(myDOM);

▼ <body>
  <h1 id="title">DOM(Document Object Model)</h1>
  <h2 id="sub_title">문서 객체 모델</h2>
  ▶ <script>...</script>
  ▶ <script>...</script>
</body>

VM316:1

▼ body ⓘ
  aLink: ""
  accessKey: ""
  ariaAtomic: null
  ariaAutoComplete: null
  ariaBrailleLabel: null
  ariaBrailleRoleDescription: null
  ariaBusy: null
  ariaChecked: null
  ariaColCount: null
  ariaColIndex: null
  ariaColSpan: null
  ariaCurrent: null
  ariaDescription: null
  ariaDisabled: null
  ariaExpanded: null
  ariaHasPopup: null
  ariaHidden: null
  ariaInvalid: null
  ariaKeyShortcuts: null
  ariaLabel: null
  ariaLevel: null
  ariaLive: null
  ariaModal: null
  ariaMultiLine: null

```

• DOM 트리

⇒ 웹 페이지의 모든 요소를 Document 객체가 관리하기 때문에 웹 페이지의 요소들을 관리하고 제어하기 위해서는 Document 객체가 웹 페이지 요소들을 잘 반영하는 자료구조를 가지고 있어야 합니다. 그래서 Document 객체 모델인 DOM은 트리 자료구조 형태를 가지고 있으며, 트리 자료 구조는 HTML 문서를 읽어 들이고 제어하기 가장 좋은 자료구조입니다.

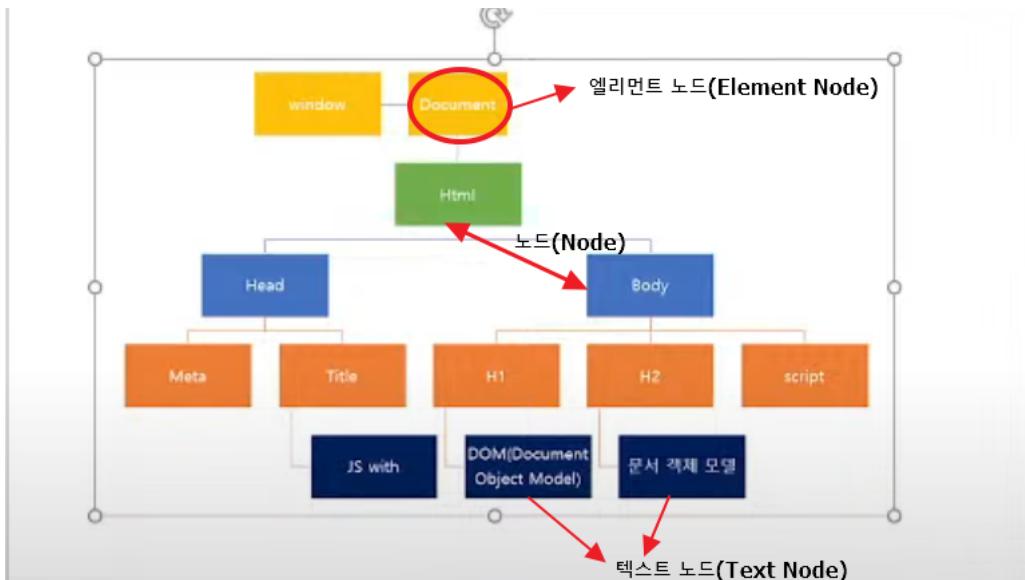
⇒ html에서 body, head 태그를 생략하여도 되지만, 두가지 태그를 이용하는 이유는 body 태그 가장 마지막 구간에 스크립트 태그를 사용하여 위 → 아래로 읽어드리는 프로그램의 특성에 따라 오류를 방지하기 위한 구분 요소로 사용하기 위함이다.

```

1  <!DOCTYPE html>
2  <html lang="en">
3      <!--<head>-->
4          <meta charset="UTF-8">
5          <title>웹컴바인4</title>
6      <!--</head>-->
7      <!--<body>-->
8          <h1 id="title">DOM(Document Object Model)</h1>
9          <h2 id="sub-title">문서 객체 모델</h2>
10         <script>
11             ...
12         </script>
13     <!--</body>-->
14 </html>
15

```

※ DOM 트리 계층구조



2. 객체의 생성방법

```

let patient = {
    name : "JiHun",
    age : 20,
    disease : 'old'
}

```

- 위의 코드는 patient라는 객체를 생성하고, 거기에 name, age, disease라는 정보를 입력하고 있다
- 이처럼 관련된 정보들을 하나로 묶어서 데이터로 저장해주는게 바로 객체라는 개념이라고 할 수 있다

- 배열 [] 과는 다른 개념

```
let 객체이름 = {
    key : value,
    key : value,
    key : value
}

ex)
let patient = {
    name: "JiHun",
    age : 20,
    disease : 'old',
    b : 19,
    c : "aaa",
    c : "ccc"
}
console.log(patient);
```

06_객체.html:69

```
{name: 'JiHun', age: 20, disease: 'old', b: 19, c: 'aaa'} i
  age: 20
  b: 19
  c: "aaa"
  disease: "old"
  name: "JiHun"
▶ [[Prototype]]: Object
```

※ 코드에 작성된 우선순위에 관계 없이 알파벳 순서로 정렬되어 출력된다

3. 객체 활용 방법

- 2번에서 생성한 객체를 활용해서 그 안에 들어있는 정보들 중 한가지만 보고싶다면 직접 접근하는 방법이 있다
- 객체이름.key의 양식으로 사용되어진다
- key와 value 또는 키와 키 값의 이름으로 사용되어지는 객체는 흔사 사전과 같은 양식으로 이루어져있다
- 배열과 혼동할 수 있지만, 배열의 방식으로 이 객체의 key 값을 불러오는 방법은 다음과 같다
⇒ 객체이름['key'] == 객체이름.키
- 객체의 내용을 수정하거나, 참조하는 방법은 배열과 유사한 방식으로 사용되어진다
⇒ 객체이름.key = 바꿀 value의 내용 or 객체이름['key'] = 바꿀 value의 내용

- 객체로 만든 정보들을 배열에 넣을 수도 있는데, 다음과 같은 방식으로 사용되어진다

⇒ let 배열이름 = [{name:"JiHun", age:20}, {name:"JiSu" , age=19}, {name:"JiMin" , age=18}]

※ 여기서 배열에 저장된 첫번째 정보만 보려면?

⇒ 배열이름[0] 이라고 하면 가능하다

※ 첫번째 정보에서 name 값만 보고 싶다면?

⇒ 배열이름[0].name or 배열이름[0]['name']

- 객체에는 단순한 값(데이터) 뿐만 아니라 함수(기능)도 넣을 수 있다
- 활용 예시

```
let patient = {
    name: "JiHun",
```

```

        age : 20,
        disease : 'old'
    }
console.log(patient.name);
console.log(patient['name']); // 배열처럼 사용
patient.name = "JiSu"; // 값 수정 #1
// patient['name'] = "JiSu"; 값 수정 #2

※ 배열 안에 객체 선언 #1
let patientList = [{name: "JiHun", age : 20}, {name: "JiSu", age : 19}, {name: "JiMin" , age : 18}];
console.log(patientList);

※ 객체 안에 배열 선언
let test1 = {
    name:["JiHun" , "JiSu" , "JiMin" , "JiYoung"],
    age:[20, 19, 18, 17]
}
console.log(test1['name'][2])
// test1.name[2] == test1['name'][2]
// => test1의 이름 중 2번째 값(JiMin) 호출

※ 배열 안에 객체 선언 #2
let test2 = [{name:"MinSu", age:20}, {name:"MinJi", age:19}, {name:"MinYoung", age:18}, {name:"MinSik" , age:17}];
console.log(test2[3]['name']);
// test2[3]['name'] == test2[3].name;
// => test2의 3번째 값의 이름값(MinSik) 호출

```

4. 주의 사항

- 배열(Array)과는 다른 존재이므로 혼동해서는 안된다
- 배열과 같은 방식으로 사용할 수 있다
- 객체 지향성 프로그래밍(Object Oriented Programming)
 - ⇒ 위에서 언급한 객체라는 것 때문에 '객체'? 지향형 프로그램이리라는 것인가 오해할 수 있지만, 위에서 언급한 객체는 자료형으로서의 객체 인 것이고, 객체지향형 프로그램에서 말하는 개념과는 다르다

※ JavaScript는 프로토타입 객체지향 언어(Prototype Object Oriented Language)이다

• 조건문

1. 조건문 if, if ~ else

- 영어 if의 뜻과 거의 흡사하다. '만약 ~라면 ~하겠다'
- if가 필요한 이유는 여러가지의 케이스마다 실행하고자 하는 내용을 달리하기 위해서 필요한 문법이다

2. if문의 사용

```

if(조건){
    조건이 참(true)이라면 실행할 코드
}

```

- 조건이 true 라면 중괄호 안으로 들어와 코드를 실행한다. 이 중괄호 영역을 if블럭 또는 if절이라고 한다
- 조건이 참(true)인지 거짓(false)인지에 따라서 실행이 될 수도 있고, 아닐 수도 있다

```

if(조건){
    조건이 참(true)일 경우 실행할 코드
} else {
    조건이 거짓(false)일 경우 실행할 코드
}

```

- 조건이 true 라면 if 다음의 중괄호 안으로 들어와서 코드를 실행하고, false 라면 else 다음의 중괄호 안으로 들어와 코드를 실행한다
- if 다음의 중괄호 영역을 if절 또는 if블럭이라고 하고, else 다음의 중괄호 영역을 else절 또는 else블럭이라고 한다
- 조건이 참(true)이라면 if절을, 거짓(false)라면 else절을 실행한다. 즉, 둘 중 하나는 반드시 실행한다

```

if(조건1){
    조건 1이 참(true)이라면 실행할 코드
} else if(조건2){
    조건 2가 참(true)이라면 실행할 코드
} else {
    조건1, 조건2가 모두 거짓(false)이라면 실행할 코드
}

```

- 조건 1이 true라면 첫번째 중괄호 안으로 들어가서 코드를 실행하고, false라면 다음 조건 2로 넘어간다
- 조건 2가 true라면 두번째 중괄호 안으로 들어가서 코드를 실행하고, false라면 넘어간다
- 조건 1, 2가 모두 false라면 else 다음의 중괄호로 들어가 실행한다

3. 조건문의 중첩사용

```

if(조건1){
    if(조건1-1){
        조건 1과 1-1이 모두 참(true)일 때 실행할 코드
    } else {
        조건 1은 참(true), 조건 1-1은 거짓(false)일 때 실행할 코드
    }
} else if(조건2){
    조건 2가 참(true)일 때 실행할 코드들
} else {
    if(조건3){
        조건1, 조건2가 모두 거짓(false), 조건3은 참(true)일 때 실행할 코드
    } else {
        조건1, 조건2, 조건3이 모두 거짓(false)일 때 실행할 코드
    }
}

```

- 조건문 안에 다른 조건문을 넣어서 여러가지 조건을 총족할 때 실행할 코드를 작성할 수 있다

※ 4일차 과제풀이

```

// 문제 1) 유저가 입력하는 숫자가 0인지, 음수인지 양수인지 판단하는 프로그램을 만들어보시오.

let num = 8;
if(num > 5){
    if(num < 10){
        console.log(num + "는 5보다는 크고 10보다는 작은 양수");
    } else {
        console.log(num + "는 10보다 큰 양수");
    }
} else if(num == 0){
    console.log(num + "는 0 입니다");
} else {
    if(num < 0){
        console.log(num + "는 음수입니다");
    } else {
        console.log(num + "는 숫자를 입력하세요")
    }
}

// 문제 2) 나는 대학교 교수다. 레포트 점수에 따라서 등급을 매기는 프로그램을 만들어보시오.

let score = 75;
let grade = ''; // 등급에 대한 정보 저장용 변수

if(90 <= score && score <= 100){
    grade = "A"
} else if(80 <= score && score <= 89){
    grade = "B"
} else if(70 <= score && score <= 79){
    grade = "C"
} else if(60 <= score && score <= 59){
    grade = "D"
} else if(0 <= score && score <= 59){
    grade = "F"
} else {
    console.log("잘못된 범위의 점수입니다 ~!")
}
console.log(grade);

```

```
// 문제 3) 한 지원자가 우리회사에 지원을 했다. 지원자가 사용가능한 스킬은 배열에 제공이 된다.
// JavaScript와 React 둘 다 할 줄 안다면 "합격!" JavaScript와 React 둘 중 하나만 할 줄 안다면 "예비!"
// 둘 다 할 줄 모른다면 "탈락!"을 보여주는 프로그램을 짜보자.

let skills = ['HTML', 'CSS', 'JavaScript', 'React'];

if(skills.includes('JavaScript') && skills.includes('React')){
    console.log("합격!");
} else if(skills.includes('JavaScript') || skills.includes('React')){
    console.log("예비!");
} else if(skills != skills.includes('JavaScript') && skills != skills.includes('React')) {
    console.log("탈락!");
} else {
    console.log(skills);
}
```

• 유사 조건문

- ⇒ 조건문과 비슷한 다른 방식들
- ⇒ 조건문 if, if ~ else, if ~ else 외 다른 표현

1. switch 문

- 문자가 주는 느낌대로 가 바꿀 것 같은 문법이다
- 조금 더 간결하고 의미가 명확해 보인다는 장점이 있음
- case 가 값으로 딱 정해진 경우에만 사용 가능
- 조건이 비교식일 경우 사용불가
- 범위를 표현하는데 있어서 불가능은 아니지만, 거의 불가능에 가깝고, 가능하다고 해도 매우 비효율적이다

```
let menu = 1;
if(menu == 1){
    console.log("물건 사기")
} else if(menu == 2){
    console.log("잔고 확인")
} else if(menu == 3){
    console.log("히스토리 확인")
} else {
    console.log("홈으로 돌아가기")
}
```

- 예를 들어 위와 같은 방식의 키오스크 프로그램이 있다고 하면 이를 switch 문으로 표현하면 다음과 같다

```
let menu = 1;
switch(menu){
    case 1:
        console.log("물건 사기")
        break
    case 2:
        console.log("잔고 확인")
        break
    case 3:
        console.log("히스토리 확인")
        break
    case 4:
        console.log("홈으로 돌아가기")
}
```

- 이와 같이 if문을 대체하여 사용할 수 있는 것이 switch 문이다
- default는 else처럼 매칭되는 case가 없을 때 실행된다
- case 마다 break를 넘겨 주는 이유는 그렇게 하지 않으면 코드의 전체를 실행하기 때문이다
- switch문으로 구현할 수 있는 코드는 모두 if문을 사용해서 구현이 가능하지만, 그 반대의 경우는 아니다
- 결론 : 지금은 잘 쓰여지지 않아 코드의 형태만 알고 넘어가도 충분함

2. 삼항연산자(조건연산자)

- switch와 다르게 제법 많이 사용되어지는 방식이다. if else를 대체할 수 있다
- 연산자이므로 연산의 결과값을 반환하게 된다. (즉, 저장할 변수와 같은 것들이 있어야 한다는 이야기)
 - ⇒ let 변수이름 = 조건 ? 조건이 true 일 때 실행 : 조건이 false일 때 실행
- 삼항연산자가 사용되어지는 경우는 다음과 같다
 - ⇒ 조건이 많지 않거나, return 이 하나만 코드에 있을 때, 반환값이 딱 하나만 코드에 있을 때
- 매우 복잡한 방식의 if문에서는 사용이 불편하다

```
let menu = 8;
if(menu <= 3){
    console.log('범위 안의 숫자입니다')
} else {
    console.log('범위 밖의 숫자입니다')
}

- 위의 if ~ else 문을 딱 한줄에 줄여서 표현이 가능하다

let answer = menu <= 3 ? '범위 안의 숫자입니다' : '범위 밖의 숫자입니다'

- 3항 연산자의 중첩

let answer = menu <= 3 ? '0초과 3이하의 숫자' : menu <= 0 ? '0이하의 숫자' : '그 외의 숫자'
```

- 이렇게 사용하는 것을 삼항연산자 또는 조건연산자라고 한다

• 반복문(for, while, do-while)

1. 개요

- 프로그래밍을 하는 근본적인 이유 : 사람이 하기 귀찮은 것들을 컴퓨터에게 시키는 것
- 반복문의 종류 : for문, while문, do ~while문
- 반복의 필수 요소 3가지 : 반복을 위한 변수선언, 반복의 범위, 반복의 종료를 위한 증감연산
- 반복의 종류
 - ⇒ 범위가 정해진 경우의 반복 : for문
 - ⇒ 범위가 정해지지 않은 경우의 반복 : while문

2. while문

- while문의 기본 구조

```
반복을 위한 변수의 선언
while (반복의 조건) {
    반복될 내용
    증감연산
}
```

- 위의 양식처럼 사용되어 진다

- while문은 보통 정확한 반복의 횟수를 알 수 없는 경우에 사용되어 진다
- 반복의 조건이 true라면 {} 안에 들어가 있는 내용을 실행하는 구조이다

3. for문

- for문의 기본 구조

```
for(초기식; 조건식; 증감식){
    반복 내용
}
```

- 위의 양식처럼 사용되어 진다

- **for문**은 보통 정확한 반복의 횟수를 알 수 있는 경우에 사용되어 진다
- 반복의 조건이나 범위의 정보만큼 증감연산 하면서 실행되어지는 구조이다

4. 반복문의 중첩

- **for문의 중첩**

```
for(let i = 0; i < 3; i++){
    for(let j = 0; j < 3; j++){
        console.log("[", i, j, "]")
    }
}
```

- **while문의 중첩**

```
let i = 0
let j = 0
while(i < 3){
    j = 0
    while(j < 3){
        console.log("[", i, j, "]")
        j++
    }
    i++
}
```

- 중첩으로 사용되어지는 반복문에서 바깥에 있는 반복문은 큰 톱니바퀴, 안쪽에 있는 반복문은 작은 톱니바퀴이다
- 작은 톱니바퀴가 한바퀴를 다 돌아야, 큰 톱니바퀴를 한칸 이동하는 방식으로 큰 톱니바퀴가 다 돌때까지 반복하는 것이 기본
- 통상적으로 3중 이상으로 중첩시켜서 사용하는 경우는 잘 없다고 봐도 무관하다
- 반복문의 중첩이 많을수록 코드를 읽어들이고 연산하는 시간이 오래 걸린다
- 그래서 반복이 중첩이 되어지면 '과연 이게 최선인가?'라는 고민을 한번쯤은 해봐야 한다

5. 반복문의 활용

- **반복문은 배열과 아주 궁합이 좋다**

- 사실 for문의 존재이유는 배열때문이라고해도 좋을만큼 궁합이 좋다
- 배열은 순서가 중요한 자료형이고, 반복문은 그 순서들에 하나씩 순차적으로 접근하는게 가능하기 때문이다
- 배열에는 length라는 데이터가 존재하는데 이는 길이값을 저장하고 있으므로, for문의 반복범위를 length 만큼 해주면 배열의 길이만큼 반복하게 된다

※ 5일차 과제풀이

```
// 문제 1) 구구단 (for문)
for(let dan = 2 ; dan < 10 ; dan++){
    console.log("<" + dan + "단 시작!!" + ">")
    for(let num = 1 ; num < 10 ; num++){
        console.log(dan + " " + "x" + " " + num + " " + "=" + " " + dan * num)
    }
    console.log("-----")
}

console.log("");

// 문제 2) 구구단 (while문)
let dan = 2;
let num = 1;

while(dan < 10){
    num = 1
    console.log("<" + dan + "단 시작!!" + ">") //log를 중간중간 찍어보면서 코드의 흐름을 이해하는게 중요함
    while(num < 10){
```

```

        console.log(dan + " " + "x" + " " + num + " " + "=" + " " + dan * num)
        num++
    }
    dan++
    console.log("-----")
}

// 문제 3) 8 'x' 8 '=' 64 의 형식이 아니라 8x8=64 의 형식으로 출력
// 쉼표(,) 연산자를 사용하면 별도의 객체로 반환
// => console.log(dan, "+", num, "=", dan * num) ==> 8 'x' 8 '=' 64

// 플러스(+) 연산자를 사용하면 문자열끼리 연결하여 새로운 문자열로 반환
// => console.log(dan + "+" + num + "=" + dan * num) ==> 8x8=64

```

※ while문과 do~while문의 차이

```

let a = 0;
while(a < 3){
    console.log(a)
    a++
}

do{
    console.log(a)
} while(a > 3)

※최초 1회는 무조건 실행하고 while문 실행

```

※ do~while 문 사용 예제

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ex02. do-while문 실질적 사용 예제</title>
</head>
<body>
    <script>
        let inputString = null

        alert("메세지를 입력하세요")
        alert("프로그램을 종료하려면 q를 입력하세요")
        // 최소 한번은 무조건 실행된다. 사용자의 의도를 물어보는 프로그램에 사용하면 된다.

        // 'q' 를 입력하기 전까지 무한 루프
        do{
            inputString = prompt("메세지를 입력하세요.")
            document.write(inputString + "<br/>")
        // 'q' 를 입력 받으면 결과값으로 true가 출력(루프 계속 진행)되며, 이 결과를 !(논리부정연산자)를 통해 false로 뒤집는다
        }while(!(inputString == "q"));

        document.write("프로그램 종료")
    </script>
</body>
</html>

```

- 중지(break) / 생략(continue)

※ 제어문을 보다 효율적으로 사용하는 방법

1. break : 중지

- 현재 내가 속해 있는 반복문 {} 를 탈출한다
- 전체적인 반복 횟수에 영향을 미친다

2. continue : 생략

- 조건검사 or 증감연산으로 이동한다
- 전체적인 반복 횟수에 영향을 미치지 않는다

3. 반복문 무한루프

- for문, while문의 경우 무한루프가 존재한다
- 말 그대로 중지되는 것 없이 계속 반복하게 되는 것을 말한다
- 반복의 조건이 true라면 무한루프라고 한다
- break 키워드와 함께 사용되어진다
- for문보다는 while에서 더 많이 사용되어진다
- 예외처리에서도 사용되어진다. (적극적인 예외처리)

```

1. break문 예시
- 1 ~ 99까지 정수 중 5와 7로 나누었을 때 나머지가 0인 정수 출력

let num = 1
let search = false

while(num < 100){
    if(((num % 5) == 0) && ((num % 7) == 0)){
        search = true
        break           * if반복문 내부 {}에 존재하더라도 반복문 전체의 {}를 벗어난다
    }
    num++
}
if(search){
    console.log("찾는 정수 : " + num)
}else {
    console.log("찾는 정수가 없습니다")
}

2. continue문 예시
- 1부터 100까지 1씩 증감하면서 5와 7로 나누었을 때 나머지가 0이 아닐 경우 continue를 실행하면서
조건 검사로 돌아가게 되고, 나머지가 0일 경우 반복문을 빠져 나오면서 count를 증감하고 출력

let number = 0
let count = 0

while(number < 100){
    number++
    if(((number % 5) != 0) || ((number % 7) != 0)){
        continue
    }
    count++
    console.log(number)
}

console.log("count : " + count)

3. 무한루프 예시
- 1부터 if의 조건을 충족할 때 까지 계속(무한) 반복

let natNum = 1
while(true){
    if(((natNum % 6) == 0) && ((natNum % 14) == 0)){
        break
    }
    natNum++
}
console.log(natNum)

```

Ex 01. 난수 맞추기 게임

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<title>Ex01. 난수를 발생시켜 맞추는 게임</title>
</head>
<body>
<script>
    let low = 0;           // 범위의 최소값
    let high = 0;          // 범위의 최고값
    let card = 0;          // 정답을 저장할 변수

    // 정답 유추를 위한 무한 루프
    while(true){
        let count = 0;      // 사용자의 시도 횟수
        low = 0;
        high = 99;
        Math.floor(Math.random() * 100) // 0 ~ 99까지의 난수 발생
        alert("수를 결정하였습니다. 맞춰보세요.")

        // 내부 루프의 경우는 정해진 난수를 맞출때까지 무한 반복
        while(true){
            let n = 0;
            alert(low + " - " + high) // 값의 범위를 출력
            count++;
            alert("시도횟수 : " + count + "번째");

            n = prompt("숫자를 입력하세요!") // 사용자로부터 숫자를 입력 받음
            n = parseInt(n) // 입력받은 숫자를 정수로 변환

            if(n > high || n < low){
                alert("값의 범위를 벗어났어요")
            } else {
                // 정상적인 범위의 수가 입력된 경우
                if(n === card){
                    alert("정답입니다. 짹짝짝!!")
                    break
                } else if(n > card){
                    alert("정답은 더 낮은 수입니다.")
                    high = n // 범위 재지정을 위한 값 대입
                } else{
                    alert("정답은 더 높은 수입니다.")
                    low = n // 범위 재지정을 위한 값 대입
                }
            }
        }
        // 내부루프 종료

        alert("시도횟수는" + count + "입니다")

        let con = confirm("다시 게임을 하시겠습니까")
        if(con == false){
            break
        }
    }
</script>
</body>
</html>

```

• 함수(function)

1. 함수란?

- 여러줄의 코드(명령)을 하나로 묶어놓은 주머니 역할
- 반복되는 기능을 주머니 안에 넣어두고 필요할 때 주머니 이름만 불러내면 기능이 실행되어진다

2. 함수 정의

```

function 함수이름(){
    함수안에 들어갈 내용
}

```

3. 함수의 사용

- 함수를 정의만 한다고 해서 기능들이 실행되어지지 않는다
- 함수를 정의하고, 기능들이 실행되게 하려면 함수를 호출해야한다

4. 함수 호출

```
함수이름()
```

5. 주의 사항

- 함수는 호이스팅의 대상이 되어진다
- 호출을 먼저하고 나중에 선언해도 문제없이 실행되어 진다

6. 함수의 종류

- 입력이 없고, 반환도 없는 함수
- 입력은 있고, 반환은 없는 함수
- 입력이 없고, 반환은 있는 함수
- 입력이 있고, 반환도 있는 함수

※ 위 4가지 종류 정도로 나눌 수 있고, **함수는 통상 기능상자의 개념**이라고 이해하면 좋다

```
1. 입력 X, 반환 X
function noInNoOut(){
    console.log("입력 없고, 반환 없고")
}

noInNoOut()
console.log("")

2. 입력 O, 반환 X
2-1) 매개변수가 1개일 경우
function oneInNoOut(number){
    console.log("함수에 입력된 값 : " + number)
}

oneInNoOut(99)

2-2) 매개변수가 2개일 경우
function twoInNoOut(num1, num2){
    console.log("함수에 입력된 두 값의 합 : " + (num1 + num2))
}
twoInNoOut(25, 30)
console.log("")

3. 입력 X, 반환 O
function noInOneOut(){
    return "입력 없고, 반환 있고"
}

console.log(noInOneOut()) // == console.log("입력 없고, 반환 있고")

4. 입력 O, 반환 O
function oneInOneOut(number){
    return number * number
}

console.log("함수에 입력된 값의 제곱 : " + oneInOneOut(7))
console.log("")
```

7. 함수 관련 용어

- **입력 : 매개변수(Parameter)와 인자(Agument)**
- **반환 : 리턴(return)**
- **스코프 : 변수를 선언하고 사용할 때 변수가 적용되는 범위**
- **익명 함수(무명 함수, anonymous function)**
 - ⇒ 이름이 없는 함수

⇒ 함수 자체가 ‘식(Expression)’이기 때문에 함수에 할당하거나 다른 함수의 매개변수로 사용할 수도 있음

- **즉시 실행 함수** : 함수를 정의함과 동시에 실행하는 함수
- **사용자 정의 함수** : 사용자가 필요한 기능을 직접 만든 함수
- **자바스크립트 코어 함수** : 자바스크립트가 기본적으로 제공하는 함수
⇒ parseInt(), Math.random() 등

8. 사용 방법에 따른 함수 종류

- **일반 함수** : 가장 일반적으로 사용한 함수
- **중첩 함수** : 함수 안에 함수가 있는 경우
⇒ 함수 내부의 지역 변수처럼 함수 내부에서만 사용 가능하며, 일반적으로 중첩함수는 이름이 없는 이벤트 리스너로 많이 활용된다

* 클릭 이벤트 리스너가 중첩 함수로 사용된 형태

```
$(document).ready(function(){
    $('#btnStart').click(function(){
        alert("안녕하세요.");
    });
});
```

* 중첩함수의 밀접도 관계

```
var a = 10;
var b = 20;
var c = 30;
function outer_func(){
    var b = 200;
    var c = 300;
    function inner_func(){
        var c = 3000;
        document.write("1. a = " + a + "<br>");
        document.write("2. b = " + b + "<br>");
        document.write("3. c = " + c + "<br>");
    }
    inner_func();
}
outer_func();

<실행결과>
1. a = 10
2. b = 200
3. c = 3000
```

⇒ 중복 코드 또는 그룹화 : 함수 내부의 커다란 기능이나 중복 코드를 내부 함수로 만들어 재사용할 때 사용 (즉, 외부 함수와 내부 함수가 아주 밀접한 관계일 때 사용하는 것이 좋다)

- **콜백 함수** : 함수 실행 결과 값을 리턴이 아닌 매개변수로 넘어온 함수를 호출해서 넘겨주는 방식

즉, 매개변수(parameter)로 인수(argument)가 아닌 함수를 받음

- **클로저 함수** : 일반적인 함수의 경우 함수 호출에 의해 함수 내부의 실행구문을 모두 실행하게 되면 함수 내부에서 만든 지역변수가 자동으로 사라지지만 어떤 경우에는 사라지지 않고 남는 경우가 존재하는데 이 현상을 클로저라고 하며, 이 현상을 일으키는 함수를 클로저 함수라 칭한다.

참조 : <https://hanamon.kr/javascript-클로저/>

★ 클로저 : 자신이 선언될 당시의 환경을 기억하는 함수

⇒ 클로저는 내부함수가 외부함수의 맥락(context)에 접근할 수 있는 것을 가리킨다

⇒ “외부로 전달”이 항상 return을 의미하는 것은 아니라는 것

⇒ 클로저는 특정 상황에서 발생하는 ‘현상’이고 함수는 이 현상이 나타나기 위한 ‘조건’에 해당한다

※ 클로저 함수 구조

```
function 외부함수명(){
    var count = 1;
```

```

    $('btn').on('click', function() {
        count++;
        alert(count);
    })

```

※ 클로저 함수 예시

```

// 클로저를 만드는 형태 1. - 중첩함수
function outerFn() {
    let x = 10;
    return function innerFn(y) { // innerFn 함수는 클로저다.
        return x = x + y;
    }
}
let a = outerFn(); // 외부함수 호출은 한번만. 이제 a 변수는 innerFn 함수를 참조한다.
a(5); // 15;
a(5); // 20;
a(5); // 25;
// 클로저를 만드는 형태 2. - 전역에 선언한 변수를 백스 안에서 함수로 정의하고 전역에서 호출
let globalFunc;
{
    let x = 10;
    globalFunc = function(y) { // globalFunc 함수는 클로저다.
        return x = x + y;
    }
}
globalFunc(5); // 15;
globalFunc(5); // 20;
globalFunc(5); // 25;

// 클로저를 만드는 형태 3. - return 없이 클로저가 발생하는 예시
(function () {
    var count = 0;
    var button = document.createElement("button");
    button.innerText ="click";
    button.addEventListener("click", function() {
        console.log(++count, "times clicked")
    })
    document.body.appendChild(button)
})();
// 별도의 외부객체인 DOM의 메서드(addEventListener)에 등록할 handler함수 내부에서 지역변수를 참조!

```

• 함수 리터럴

```

let a = function() {
}

```

리터럴 방식	일반적인 방식과 객체 방식
<pre> var hello = function(name){ alert(name + "님 환영합니다."); } hello("honggildong"); </pre>	<pre> //일반적인 방식 function hello(name){ alert(name + "님 환영합니다."); } hello("honggildong"); //객체 방식 var hello2 = new Function("name", "alert(name+'님 환영합니다.');"); hello2("honggildong"); </pre>

• return 키워드의 2가지 기능

1) return 키워드 오른쪽에 오는 값을 함수 호출위치로 반환

2) 함수의 실행 종료

※ break와 return의 차이

- break의 경우 내가 속해있는 반복문 {}를 빠져나오지만, return의 경우 함수의 사용이 종료되기 때문에 함수 내부에 존재하는 반복문을 빠져나오는 개념이 아닌 실행 자체를 하지 않는다
- 단, 함수 내부에 다른 함수가 존재할 경우 return을 만난 함수만 사용이 종료된다

※ 함수 예시 코드 #1

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>함수의 종류</title>
</head>
<body>
    <button id="btnStart">시작</button>
    <script>
        // 1. 함수 리터럴

        // 변수에 함수를 저장하는 경우
        function hi(name){
            document.write(name + "님 환영합니다.");
        }
        hi("홍길동");

        var fhi = hi;           // 함수 리터럴
        fhi("임꺽정");

        // 변수에 익명함수를 저장하는 경우 (변수가 함수명처럼 사용되는 느낌)
        var func = function(){
            document.write("함수 리터럴");
        }
        func();

        // 2. 함수를 매개변수로 이용
        function hi1(){
            alert("안녕하세요");
        }

        function hi2(){
            alert("반갑습니다");
        }

        function exec(func){      // func = hi1; , func = hi2; 와 동일
            func();
        }

        exec(hi1);
        exec(hi2);

        // 3. return 값으로 함수 이용
        $(document).ready(function(){
            function hi(){
                document.write("안녕하세요. 반갑습니다. !");
            }
            // 버튼 클릭 시 hi 함수 실행
            $('#btnStart').click(hi);
        })
    
```

// 4. 객체방식의 함수 정의 (실무에서 잘 사용되지 않는다)

```
var hi = new Function("name", "document.write(name + '님 환영합니다.')");
hi("홍길동")

// 익명함수의 확장으로 함수 정의
// 재사용 목적이 아닌 함수끼리의 충돌을 방지하기 위함
(function(name){
    document.write(name + "님 환영합니다.");
})(("홍길동"))
</script>
</body>
</html>
```

※ 함수 예시 코드 #2

* 햄버거 만들기

```
function makeHamburger(type){
```

```

        console.log("빵 깔기")
        console.log("상추 올리기")
        if(type == "불고기"){
            console.log("불고기 패티 올리기")
        } else if (type == "새우"){
            console.log("새우 패티 올리기")
        }
        console.log("토마토 올리기")
        console.log("치즈 올리기")
        console.log("빵 덮기")
    }

    function serveCoke(){
        console.log("콜라통 선택")
        console.log("얼음 담기")
        console.log("콜라 담기")
    }
    function serveFrenchFries(){
        console.log("감튀통 선택")
        console.log("감튀 담기")
    }
    makeHamburger("새우")
    serveCoke()
    serveFrenchFries()

    * 매개변수 여러개 사용
    function make_hamburger(type, size, num){
        console.log("버거의 종류는?", type)
        console.log("빵 깔기")
        if(type == "불고기"){
            console.log("불고기 패티 올리기")
        } else if (type == "새우"){
            console.log("새우 패티 올리기")
        } else if (type == "치킨"){
            console.log("치킨 패티 올리기")
        }
        console.log("상추 올리기")
        console.log("토마토 올리기")
        console.log("치즈 올리기")
        console.log("빵 덮기")
        console.log(type+"버거", size+"사이즈", num+"개")
    }

    make_hamburger("치킨", "L", 3)

    * 다중 함수 이용하기
    function makeSet(){
        makeHamburger("새우")
        return
        serveCoke()
        serveFrenchFries()
    }

    makeSet()

```

Ex) 중첩함수로 만든 사칙연산

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ex00. 중첩함수로 되어진 사칙연산</title>
</head>
<body>
    <script>
        function calculator(op, num1, num2){
            var result = " ";

            switch(op){
                case "+":
                    result = add(num1, num2);
                    break;
                case "-":
                    result = sub(num1, num2);
                    break;
                case "*":
                    result = mul(num1, num2);
                    break;
                case "/":
                    result = div(num1, num2);
            }
        }
    </script>

```

```

        break;
    default :
        result = "지원하지 않는 연산자입니다."
    }
    // 위 switch에서 사용한 함수를 중첩함수 형태로 구현
    function add(num1, num2){
        return num1 + num2;
    }

    function sub(num1, num2){
        return num1 - num2;
    }

    function mul(num1, num2){
        return num1 * num2;
    }

    function div(num1, num2){
        return num1 / num2;
    }

    // 외부함수의 리턴문
    return result;
}

document.write("결과 : " + calculator("+", 100, 200), + "<br/> ")
document.write("결과 : " + calculator("-", 100, 200), + "<br/> ")
document.write("결과 : " + calculator("*", 100, 200), + "<br/> ")
document.write("결과 : " + calculator("/", 100, 200), + "<br/> ")
</script>
</body>
</html>

```

※ 6일차 과제 풀이

```

// 문제 1) 1부터 100까지 더하는 for문을 만들고 그 결과를 출력해보자.

1. for문
let sum = 0;

for(let num = 1 ; num < 101 ; num++){
    console.log(num + " " + "+" + " " + sum + " " + "=" + " " + (sum + num))
    sum += num
}
console.log("1부터 100까지의 합 : " + sum)

2. while문

let num1 = 1
let sum1 = 0
while(num1 < 101){
    sum1 += num1
    num1++
}
console.log(sum1)

// 문제 2) 1부터 100까지 홀수만 출력해보자.

1.
for(let num = 1 ; num < 101 ; num++){
    if((num % 2) == 1){
        console.log(num)
    }
}

2.
for(let num2 = 1 ; num2 < 101 ; num2 = num2 + 2){
    console.log(num2)
}

// 문제 3) 1부터 50까지 369의 결과를 프린트해보자.
for(let i = 1; i < 51 ; i++){
    let stringValue = i.toString()          // 변수 stringValue에 대입되는 i(1~50) 값을 문자열로 변경
    let result = ""                         // res에 공백 선언 (result.length = 0)
    for(let j = 0; j < stringValue.length; j++){
        if(stringValue[j] == "3" || stringValue[j] == "6" || stringValue[j] == "9"){
            result += "짝"                  // stringValue[0 ~ 1] 값이 "3", "6", "9" 일때 result에 "짝" 대입 ※33의 경우 0번째와 1
        }
    }
    console.log(result.length > 0 ? result : i) // result = ""(공백)이 아니면 "짝" "(공백)이면 i값을 넣어 출력 ※위 for문으로 result에 3,
    // 6, 9가 들어간 경우 "짝" 을, 그 외 해당 문자열(i) 출력
}

```

```

        }

// 문제 4) 주어진 숫자가 소수이면 true 아니면 false를 출력하는 프로그램을 짜세요.
let prime_num = 13
let isPrime = true

if(prime_num === 1){
    isPrime = false
}
for(let i = 2; i < prime_num; i++){
    if(prime_num % i ==0){
        isPrime = false
    }
}
if(isPrime){
    console.log("소수입니다")
} else {
    console.log("소수아닙니다")
}

```

※ Java 와 JavaScript 나눗셈(/) 연산 결과 비교

1. 369게임 알고리즘 코드 비교
 - 아래의 코드에서 결과값 비교 시 Java 코드의 경우 30 ~ 39까지 "짝"이라는 결과가 출력되지만, JavaScript의 경우 30 ~ 39중 1의 자리가 3, 6, 9인 숫자와 해당 숫자를 10으로 나눴을 때 3으로 떨어지는 숫자 30만 "짝"이라는 결과로 출력되는 것으로 확인

```

<Java>
public static void main(String[] args) {

    for(int i=1 ; i < 51 ; i++) {
        if((i & 10) == 3 || (i % 10) == 6 || (i % 10) ==9) {
            System.out.println("짝");
            continue;
        }if((i / 10) == 3 || (i / 10) == 6 || (i / 10) == 9) {
            System.out.println("짝");
            continue;
        }
        System.out.println(i);
    }
}

<JavaScript>
for(let num = 1 ; num < 51 ; num++){

    if((num % 10) == 3 || (num % 10) == 6 || (num % 10) == 9){
        console.log("짝")
        continue
    }
    if((num / 10) == 3 || (num / 10) == 6 || (num / 10) == 9){
        console.log("짝")
        continue
    }
    console.log(num)
}

```

2. 단순 나눗셈 연산 및 if문을 통한 결과값 비교
 - 아래의 코드에서 Java 코드의 경우 int형(정수)으로 선언된 k의 연산결과는 3(정수)이지만, JavaScript 코드의 경우 k의 연산결과가 3.1(실수) 형태로 출력되는 것으로 확인

- 각각의 k의 값으로 if문을 통한 조건 결과 출력 시 Java 코드의 경우 조건이 true(참)임으로 "몫은 3" 이라는 결과가 출력되지만, JavaScript의 경우 조건이 false(거짓)임으로 else가 없을 경우 아무것도 실행하지 않는 것으로 확인되었으며, else가 있을 경우 else 안의 내용 "왜 실행안하니?" 가 출력되는 것으로 확인 (*K == 3.1로 비교시 true(참)로 "몫은 3.1"이란 결과 출력되는 것으로 확인)

```

<Java>
int j = 31;
int k = j / 10;
System.out.println(k);

if(k == 3) {
    System.out.println("몫은" + k);

<JavaScript>
let j = 31;
let k = j / 10;
console.log(k)

if(k == 3){
    console.log("몫은" + k)
} else{
    console.log("왜 실행안하니?")
}

```

The screenshot shows an IDE interface with several windows:

- Java Editor:** Displays a Java file named `Three_3.java` with code that prints numbers from 1 to 50, filtering for multiples of 3, 5, and 7.
- Console Tab:** Shows the output of the Java application, which is the same as the browser's console output.
- File Explorer:** Shows a directory structure with files named 01_bansu.html through 07_zeru.html.
- JavaScript Editor:** Displays a script with various conditional statements and loops, including a prime number check and a template literal example.
- Console Tab (Browser):** Shows the browser's developer tools console tab with log outputs corresponding to the script's execution.

※ 별탈 세우기

1. 원쪽정렬

```
console.log("별탈")
let star = ''
for(let i = 1; i < 6; i++){
    star += '*'
    console.log(star)
}
```

※ Template literals

* ``(백틱)을 활용한 템플릿 리터럴

```
let year = 2023
let month = 4
let day = 21

console.log(`오늘은 ${year}년 ${month}월, ${day}일`)
console.log(`오늘은 ${year}년, ${month}월, ${day}일`)
```

```
const city = "seoul";
const name = "jim"
const age = 21;

// 일반적인 문자열
console.log("저는 " + city + "에 살고 있구요, 이름은 " + name + ", 나이는 " + age + " 살입니다.");

// () 템플릿 리터럴
console.log(`저는 ${city}에 살고 있구요, 이름은 ${name}, 나이는 ${age} 살입니다.');
```

※ 7일차 과제풀이

```
// 문제 1) meetAt 함수를 만들어주세요.

// (조건) 인자 3개
// 1) 첫번째 인자는 년도에 해당하는 숫자입니다.
// 2) 두번째 인자는 월에 해당하는 숫자입니다.
// 3) 세번째 인자는 일에 해당하는 숫자입니다.

// (결과)
// 1) 년도 인자만 받았을 경우 → "1234년" 과 같은 형식의 문자열을 리턴 해주세요.
// 2) 년도, 월 인자를 받았을 경우 → 년도와 월을 조합해서 "1234년 5월" 과 같은 형식의 문자열을 리턴 해주세요.
// 3) 년도, 월, 일 인자를 전부 받았을 경우 → 년도, 월, 일을 조합해서 "1234/5/6" 과 같은 형식의 문자열을 리턴 해주세요.

1.
function meeAt(year, month, day){
    if(day){
        return year + "/" + month + "/" + day
    }else if(month){
        return year + "년" + month + "월"
    }else if(year){
        return year + "년"
    }
}

console.log(meeAt(2023))
console.log(meeAt(2023, 4))
console.log(meeAt(2023, 4 , 20))

2.
function meeAt(year, month, day){
    let todayYear = year
    let todayMonth = month
    let todayDate = day

    if(todayDate){
        return `${todayYear}/${todayMonth}/${todayDate}`
    }
    if(todayMonth){
        return `${todayYear}년 ${todayMonth}월`
    }
    if(todayYear){
        return `${todayYear}년`
    }
}

console.log(meeAt(2021))
console.log(meeAt(2021, 11))
console.log(meeAt(2021 , 4, 21))

* 1번과 2번의 차이점은 1번의 경우 else가 존재하기 때문에 else의 조건까지 모두 검사 하는 반면 (* else는 단독실행 불가)
  2번의 경우 조건에 부합하는 if문만 단독으로 실행하기 때문에 불필요한 검사 과정을 생략할 수 있다

// 문제 2) findSmallestElement 함수를 구현해 주세요.

// (조건) findSmallestElement 의 arr 인자는 숫자 값으로만 이루어진 배열입니다.
//         이용되는 배열
//         [100,200,3,0,2,1]

// (결과)
// 1) arr 의 값들 중 가장 작은 값을 리턴 해주세요.
// 2) 만일 arr 가 비어있으면 0을 리턴 해주세요.
// 3) 예를 들어, 다음과 같은 배열이 인자(input)으로 들어왔다면 0이 리턴 되어야 합니다.

function findSmallestElement(arr){
    let min = arr[0] // arr[0]번째 값이 저장되어있기 때문에 for문에서 1번째부터 검사함
    for(let i = 0 ; i <= arr.length; i++){
        if(min > arr[i]){
            min = arr[i]
        }
    }
    return min
}

let arr = [100,200,3,0,2,1]
console.log(findSmallestElement(arr))

// 문제 3) 돈을 매개변수로 받으면 몇개의 지폐와 동전이 필요한지 최소 개수를 계산해주는 함수를 만드시오

// (결과 예시)
// 제공받은 돈이 12300인 경우
// 50000 X 0
// 10000 X 1
```

```

// 5000 X 0
// 1000 X 2
// 500 X 0
// 100 X 3

1.
function calMoney(don){
    console.log("받은 돈 : " + don + "원")

    let oman = don / 50000
    let man = (don % 50000) / 10000
    let ochun = (don % 10000) / 5000
    let chun = (don % 5000) / 1000
    let obak = (don % 1000) / 500
    let bak = (don % 500) / 100

    console.log("5만원" + "X" + " " + Math.floor(oman))
    console.log("1만원" + "X" + " " + Math.floor(man))
    console.log("5천원" + "X" + " " + Math.floor(ochun))
    console.log("1천원" + "X" + " " + Math.floor(chun))
    console.log("5백원" + "X" + " " + Math.floor(obak))
    console.log("백원" + "X" + " " + Math.floor(bak))
}

calMoney(12300)

2.
let unit = [50000, 10000, 5000, 1000, 500, 100]
function changeMoney(money){
    for(let i = 0; i < unit.length; i++){           // 1) 제공받은 돈을 unit의 0번째 값부터 순차적으로 나눈 뒤를 소수점 아래에 대입
        let num = Math.floor(money / unit[i])         // 2) 제공받은 금액에서 1)에서 대입한 num의 값과 unit의 0번째 값부터 순차적으로 곱하기 연산결과를 빼는
        money = money - unit[i] * num                // ex) num = 12300 / unit = [50000, 10000, 5000, 1000, 500, 100]           //     money = 12300 - (unit = [50000, 10000, 5000, 1000, 500, 100] * 0 ~ 5(unit[0])
    }
}

changeMoney(24600)

```

- **Git-Hub**

※ git-hub vscode 연결 순서

- 1) 구글 → **github** 검색 → 회원 가입 및 이메일 계정 로그인
- 2) 'New' 버튼을 누르고 신규 **Repositories** 생성
- 3) 구글 → git 검색 → 다운로드 및 **git bash** 실행
- 4) 아래의 명령 순차적 실행 (이름 및 메일 주소 등록 상태 확인)
 - git config --global user.name "BaeSangWon"
 - git config --global user.email "sjqcl3310@naver.com"
 - git config --list
- 5) vs code → 터미널 → 새터미널 실행 후 우측 + 버튼 → Git Bash 추가

The screenshot shows the Visual Studio Code interface. At the top, there's a dark-themed header with the Microsoft logo and the word "Visual Studio Code". Below the header is a tab bar with "문제", "출력", "디버그 콘솔", and "터미널" (Terminal). The "터미널" tab is selected and highlighted in blue. In the main area, a terminal window displays the following command-line session:

```
user@DESKTOP-757B2J1 MINGW64 ~/Desktop/js
$ git init
Initialized empty Git repository in C:/Users/user/Desktop/js/.git/
user@DESKTOP-757B2J1 MINGW64 ~/Desktop/js (master)
$ git add .

user@DESKTOP-757B2J1 MINGW64 ~/Desktop/js (master)
$ git status
On branch master
```

To the right of the terminal, a context menu is open over the terminal tab. The menu items are: PowerShell, Git Bash (which is highlighted in blue), Command Prompt, JavaScript 디버그 터미널, 터미널 분할 >, 터미널 설정 구성, 기본 프로필 선택, 작업 실행..., and 작업 구성... . Below the menu, there's a toolbar with icons for "powershell" and "bash".

6) Git Bash 에서 아래의 명령 순차적 실행

- git init
 - git add .
 - git status
 - git commit -m "First commit"
 - git remote add origin <https://github.com/BaeSangWon/test.git>
 - git remote -v
 - git push origin master
-
- **JavaScript에서의 요소(Element) 선택**

※ 자바스크립트에서의 자식 요소 선택

```

> const myTag = document.querySelector('#content');
< undefined
> console.log(myTag.children);
VM702:1
HTMLCollection(4) [h2#title-1, ul#list-1,
h2#title-2, ul#list-2, title-1: h2#title-1, lis
t-1: ul#list-1, title-2: h2#title-2, list-2:
ul#list-2] 4
▶ 0: h2#title-1
▶ 1: ul#list-1
▶ 2: h2#title-2
▶ 3: ul#list-2
▶ list-1: ul#list-1
▶ list-2: ul#list-2
▶ title-1: h2#title-1
▶ title-2: h2#title-2
length: 4
▶ [[Prototype]]: HTMLCollection
< undefined
>

```

```

5      <title>웹컴바인4</title>
6      </head>
7      <body>
8      <div id="content">
9          <h2 id="title-1">과일-1</h2>
10         <ul id="list-1">
11             <li>사과</li>
12             <li>배</li>
13             <li>한라봉</li>
14             <li>키위</li>
15             <li>토마토</li>
16             <li>딸기</li>
17             <li>대추</li>
18             <li>감</li>
19         </ul>
20         <h2 id="title-2">과일-2</h2>
21         <ul id="list-2">
22             <li>감귤</li>
23             <li>참다래</li>
24             <li>매실</li>
25             <li>수박</li>
26             <li>참외</li>
27             <li>블루베리</li>
28             <li>체리</li>
29             <li>바나나</li>

```

- **getElementsBy** : 버튼과 같이 단일 항목 하나를 가져올 때 사용, 동일항목이 존재하면 최상단에 1개만 출력
- **querySelector** : 여러개의 항목을 가져올 때 사용

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div class="test">안녕하세요!!</div>
    <div class="test">안녕하세요!!</div>
    <div class="test">안녕하세요!!</div>
    <div class="test">안녕하세요!!</div>
    <script>
        let getElementClass = document.getElementsByClassName('test')
        let querySelect = document.querySelector('.test')
        let querySelectAll = document.querySelectorAll('.test')

        console.log(getElementClass)
        console.log(querySelect)
        console.log(querySelectAll)
    </script>
</body>
</html>

```

```

1 <!DOCTYPE html>
2 <html lang="ko">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10  <div class="test">안녕하세요!!</div>
11  <div class="test">안녕하세요!!</div>
12  <div class="test">안녕하세요!!</div>
13  <div class="test">안녕하세요!!</div>
14 <script>
15   let getElementClass = document.getElementsByClassName('test')
16   let querySelect = document.querySelector('.test')
17   let querySelectAll = document.querySelectorAll('.test')
18
19   console.log(getElementClass)
20   console.log(querySelect)
21   console.log(querySelectAll)
22
23 </script>
24 </body>
25 </html>

```

The DevTools console output shows:

```

HTMLCollection(4) 
  ▷ 0: div.test
  ▷ 1: div.test
  ▷ 2: div.test
  ▷ 3: div.test
  length: 4
  [[Prototype]]: HTMLCollection

<div class="test">안녕하세요!!</div>

NodeList(4) 
  ▷ 0: div.test
  ▷ 1: div.test
  ▷ 2: div.test
  ▷ 3: div.test
  length: 4
  [[Prototype]]: NodeList

```

- **element.classList.add('name')**

⇒ 요소에 클래스 이름 추가

⇒ element.className += '이름'; 과 달리 스페이스를 추가해주지 않아도 됨

```

// class 이름 읽기
function getClassName() {
  alert(document.getElementById('ex').classList);
}

// class item 개수 확인
function getCountClassItem() {

  alert(document.getElementById('ex').classList.length);
}

// class 이름 하나씩 읽기
function getClassItemName() {
  const classLength =
    document.getElementById('ex').classList.length;
  for(let i = 0; i < classLength; i++){
    alert(document.getElementById('ex').classList.item(i))
  }
}

```

The Result panel shows three buttons: 'class 이름 읽기', 'class item 개수 확인', and 'class 이름 하나씩 읽기'.

- **element.classList.replace('변경전이름', '변경후이름');**

⇒ 요소 클래스 이름 변경

The 'Result' panel shows two buttons: '클래스 이름 설정/변경' and '클래스 이름 추가'.

Below the editor are buttons for 'Resources', '1x', '0.5x', '0.25x', and 'Rerun'.

The 'Result' panel is empty in this screenshot.

Below the editor are buttons for 'Resources', '1x', '0.5x', '0.25x', and 'Rerun'.

※ getElementById 와 querySelector 차이점

- **index.html**

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>javaScript</title>
</head>
<body>
    <h2>진행과정</h2>
    <ul id="doing-list">
        <li>인터랙티브 자바스크립트</li>
        <li>프로그래밍 언어</li>
        <li>프로그래머의 세계</li>
        <li>소프트웨어 이해하기</li>
    </ul>

    <h2>원료과정</h2>
    <ul id="done-list">
        <li>프로그래밍 시작하기</li>
        <li>프로그래밍 핵심개념</li>
        <li>프로그래밍과 데이터</li>
        <li>HTML/CSS 시작</li>
        <li>HTML/CSS 핵심</li>
        <li>반응형 웹퍼블리싱</li>
    </ul>
</body>
</html>
```

- common.js

```
let stand = document.querySelector('body')

// ex1) li('인터랙티브 자바스크립트') 선택방법

/* ※ <h2>진행과정</h2> : body의 children[0] */

// 1) body > #doing-list > li:firstchild의 형제요소 노드 중 이전(좌측) 요소 선택 ('인터랙티브 자바스크립트' 이전 요소가 존재하지 않음)
let a = stand.children[1].children[0].nextElementSibling.previousElementSibling
console.log(a)

// 2) body > #doing-list > li:firstchild 선택
let b = stand.children[1].children[0];
console.log(b)

// 3) body h2:nth-child(2)의 형제요소 노드 중 이전(좌측) 요소 중 0번째
let c = stand.children[2].previousElementSibling.children[0]
console.log(c)

// 4) body h2:firstchild의 형제요소 노드 중 다음(우측) 요소 중 첫번째
let d = stand.firstChild.nextElementSibling.firstChild
console.log(d)

// ex2) li('소프트웨어 이해하기') 선택방법
let doingList = document.querySelector('#doing-list')
let doneList = document.querySelector('#done-list')

// 1) #doingList의 마지막 자식 요소
let e = doingList.lastElementChild
console.log(e)

// 2) #doingList의 3번째 자식 요소
let f = doingList.children[3]
```

- 요소 노드 프로퍼티

⇒ element.children : 자식 요소 노드(엘리먼트의 자식 요소 모음, HTMLCollection)
⇒ element.firstElementChild : 자식 요소 노드(엘리먼트의 첫 번째 자식 요소 하나)
⇒ element.lastElementChild : 자식 요소 노드(엘리먼트의 마지막 자식 요소)
⇒ element.parentElement : 부모 요소 노드(엘리먼트의 부모 요소 하나)
⇒ element.previousElementSibling : 형제 요소 노드(엘리먼트의 이전 혹은 좌측에 있는 요소 하나)
⇒ element.nextElementSibling : 형제 요소 노드(엘리먼트의 다음 혹은 우측에 있는 요소 하나)

- 모든 노드에 대한 이동 프로퍼티

⇒ node.childNodes : 자식 노드(node의 자식 노드 모음 NodeList)
⇒ node.firstChild : 자식 노드(node의 첫 번째 자식 노드 하나)
⇒ node.lastChild : 자식 노드(node의 마지막 자식 노드 하나)
⇒ node.previousSibling : 형제 노드(node의 이전 혹은 좌측에 있는 노드 하나)
⇒ node.nextSibling : 형제 노드(node의 다음 혹은 우측에 있는 노드 하나)

- 추가 / 삭제 관련

- ⇒ **text명령 , html명령**

⇒ HTML 태그에 있는 내용을 변경하거나 저장하는 명령

- **innerHTML** : HTML 코드를 문자열로 리턴(내부의 줄바꿈 등 모든 요소 포함), 내부 HTML을 수정할 때 주로 사용 (**내부의 값을 완전히 새로운 값으로 교체하기 때문에 사용에 주의가 필요함**)

```
<!DOCTYPE html>
<html lang="ko">
<head>
```

```

<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>javaScript</title>
</head>
<body>
    <h2>진행과정</h2>
    <ul id="doing-list">
        <li>인터랙티브 자바스크립트</li>
        <li>프로그래밍 언어</li>
        <li>프로그래머의 세계</li>
        <li>소프트웨어 이해하기</li>
    </ul>

    <h2>완료과정</h2>
    <ul id="done-list">
        <li>프로그래밍 시작하기</li>
        <li>프로그래밍 핵심개념</li>
        <li>프로그래밍과 데이터</li>
        <li>HTML/CSS 시작</li>
        <li>HTML/CSS 핵심</li>
        <li>반응형 웹퍼블리싱</li>
    </ul>
<script>
    let myTag = document.querySelector('#done-list')
    console.log(myTag.innerHTML)

```

요소.html:29

```

<li>프로그래밍 시작하기</li>
<li>프로그래밍 핵심개념</li>
<li>프로그래밍과 데이터</li>
<li>HTML/CSS 시작</li>
<li>HTML/CSS 핵심</li>
<li>반응형 웹퍼블리싱</li>

```

```

myTag.innerHTML = '<li>프로그래밍 핵심개념</li>'
console.log(myTag)

※ myTag에 '<li>프로그래밍 핵심개념</li>'를 대입함과 동시에 '<li>프로그래밍 핵심개념</li>' 다음 요소는 모두 삭제되어 버린다

```

요소.html:29

```

<li>프로그래밍 시작하기</li>
<li>프로그래밍 핵심개념</li>
<li>프로그래밍과 데이터</li>
<li>HTML/CSS 시작</li>
<li>HTML/CSS 핵심</li>
<li>반응형 웹퍼블리싱</li>

```

```

> console.log(myTag)
VM4078:1
▼<ul id="done-list">
  ▼<li>
    ::marker
    "프로그래밍 핵심개념"
  </li>
  ▼<li>
    ::marker
  </li>
</ul>
<+ undefined
>

```

```
myTag.innerHTML += '<li>자바스크립트 기초개념</li>'  
console.log(myTag)
```

※ 요소를 추가하기 위해서는 '복합대입연산자'를 활용

```

요소.html:35
▼<ul id="done-list">
  ▼<li>
    ::marker
    "프로그래밍 핵심개념"
  </li>
  ▶<li>...</li>
  ▼<li>
    ::marker
    "자바스크립트 기초개념"
  </li>
  ▶<li>...</li>
</ul>

```

⇒ **outerHTML** : 요소 노드 전체적인 HTML 코드를 문자열로 리턴

```
let myTag = document.querySelector('#done-list')
console.log(myTag.innerHTML)
console.log(myTag.outerHTML)
```

※ **innerHTML** 과 **outerHTML**의 차이

- **outerHTML**의 경우 감싸고 있는 요소까지 리턴

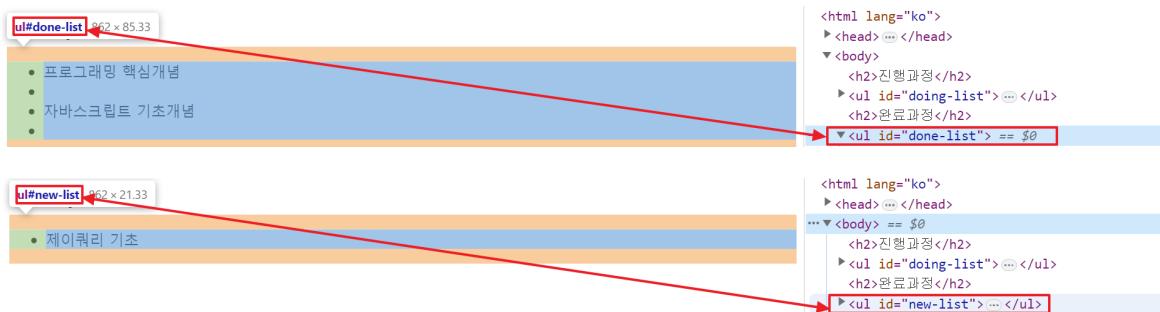
```
요소.html:29
<li>프로그래밍 시작하기</li>
<li>프로그래밍 핵심개념</li>
<li>프로그래밍과 데이터</li>
<li>HTML/CSS 시작</li>
<li>HTML/CSS 핵심</li>
<li>반응형 웹퍼블리싱</li>
```

```
요소.html:30
<ul id="done-list">
    <li>프로그래밍 시작하기</li>
    <li>프로그래밍 핵심개념</li>
    <li>프로그래밍과 데이터</li>
    <li>HTML/CSS 시작</li>
    <li>HTML/CSS 핵심</li>
    <li>반응형 웹퍼블리싱</li>
</ul>
```

```
<ul id="done-list">
    <li>프로그래밍 시작하기</li>
    <li>프로그래밍 핵심개념</li>
    <li>프로그래밍과 데이터</li>
    <li>HTML/CSS 시작</li>
    <li>HTML/CSS 핵심</li>
    <li>반응형 웹퍼블리싱</li>
</ul>

myTag.outerHTML = '<ul id = "new-list"><li>제이쿼리 기초</li></ul>'
```

※ 기존 ul의 id 값이 #done-list → #new-list로, li 값 역시 바뀐 것으로 확인할 수 있음



⇒ text 명령은 텍스트 형태의 자료만 컨트롤 할 수 있고, html 명령은 태그도 컨트롤 할 수 있다

- **textContent** : 노드(HTML문서)의 text 값을 반환 , 요소 안의 내용들 중에서 HTML 태그 부분은 제외하고 텍스트 영역만 리턴 전체를 텍스트 처리(공백, 띄워쓰기 포함)
- **innerText** : 노드의 text 값을 반환, textContent와 비슷하지만 차이점이 있다
- **textContent** 는 모든 요소를 반환하는 반면 innerText는 사람이 읽을 수 있는 요소만 가져온다 (글자사이에 공백이 많다면 textContent는 그대로 가져오지만 innerText는 한칸만 가지고 온다)

```
<div id="text"><span>hello </span> world</div>

1. hello world 사이의 모든 공백을 포함하여 결과 출력
let text = document.getElementById('text').textContent
console.log(text)

2. hello world 사이의 한칸의 공백외 중복되는 공백은 제외하고 결과 출력
let text2 = document.getElementById('text').innerText
```

```

console.log(text2)

3. hello world 에 적용되어 있는 내부 태그 및 스타일 요소 모두 출력
let text3 = document.getElementById('text').innerHTML
console.log(text3)

```



• 요소 추가하는 방법 3단계

- 1) 요소 노드 생성
- 2) 요소 노드 꾸미기
- 3) 요소 노드 추가
 ⇒ **prepend** : 메소드를 호출한 제일 첫번째 노드로 삽입
 ⇒ **append** : 메소드를 호출한 가장 마지막 노드로 삽입

```

<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>생활계획표</title>
</head>
<body>
  <h1>오늘 생활계획표</h1>
  <ol id="today">
    <li>친구와 게임하기</li>
    <li>쇼핑 하러 가기</li>
    <li>맛있는 저녁 만들어 먹기</li>
  </ol>
  <h1>내일 생활계획표</h1>
  <ol id="tomorrow">
    <li>놀이동산 가기</li>
    <li>독서 하기</li>
    <li>청소 하기</li>
  </ol>

  <div id="element">이곳은 DIV 엘리먼트입니다.</div>
</body>
</html>

```

```

let today = document.querySelector('#today');

// 요소의 첫번째에 추가
today.innerHTML = '<li>처음</li>' + today.innerHTML;

// 요소의 마지막에 추가
today.innerHTML += '<li>마지막</li>';

// 선택된 태그 외부(바깥쪽)에 추가 (즉, h1과 today(ol) 태그 사이에 요소 추가)
today.outerHTML = '<p>이전</p>' + today.outerHTML;

// 오류구문 : outerHTML의 경우 최초 실행 시 원본을 완전히 바꾸기 때문에 후속 실행되는 구문에 오류가 발생할 수 있다
today.outerHTML = today.outerHTML + '<p>마지막</p>'

// 오류구문 해결 방법
let newToday = document.querySelector('#today');
newToday.outerHTML = today.outerHTML + '<p>마지막</p>'

let tomorrow = document.querySelector('#tomorrow');

// 1) 요소 노드 생성
let first = document.createElement('li')

// 2) 요소 노드 꾸미기

```

```

first.textContent = '처음';

// 3) 요소 노드 추가
tomorrow.prepend(first);

let last = document.createElement('li');
last.textContent = '마지막';
tomorrow.append(last);

let prev = document.createElement('p')
prev.textContent = '이전'
tomorrow.before(prev)

let next = document.createElement('p')
next.textContent = '다음'
tomorrow.after(next)

// 4) 요소 노드 삭제
tomorrow.children[2].remove();

// 5) 요소 노드 이동
// prepend, append, before, after
let today = document.querySelector('#today');

today.append(tomorrow.children[1]);

// 오늘생활계획표(today) 의 2번째 자식요소(쇼핑하러가기)를 내일 계획표(tomorrow) 2번째 자식요소로 넣기
tomorrow.children[1].after('#today').children[1];

tomorrow.lastElementChild.before('#today'.children[1]);

let element = document.getElementById('element');
// 클래스 추가
element.classList.add('active');

// 클래스 삭제
element.classList.remove('active');

// 선택요소에 'active' 클래스 존재 여부 확인 후 없다면 클래스 추가
element.classList.toggle('active')

// 선택요소에 'active' 클래스 존재 여부 확인 후 있다면 true, 없다면 false 반환
element.classList.contains('active')

```

- **alert(document.getElementById('box-5').classList.item(number))**

⇒ 알람창 팝업

- **window.getComputedStyle(element)**

⇒ CSS 속성 요소 객체에 접근할 수 있도록 해주는 메서드

Ex01. 오늘 계획(To-do-list) 표 예제

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>오늘계획표</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="wrap">
        <h2 class="title">오늘 계획</h2>
        <ul id="to-do-list"></ul>
    </div>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="js/style.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "UTF-8";

body{
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
}

.wrap{
    width: 300px;
    margin: 40px;
    background-color: #fcfcfc;
    box-shadow: -5px -5px 20px #fff, 5px 5px 20px #babecc;
    border-radius: 8px;
    padding: 30px 0;
    text-align: center;
}

.title{
    color: #ff6eed;
    font-size: 30px;
    font-weight: bold;
    margin: 15px auto;
}

#to-do-list{
    width: 200px;
    margin: 0 auto 15px;
    padding: 0;
    list-style: none;
}

#to-do-list li{
    display: flex;
    align-items: center;
    justify-content: center;
    width: 90%;
    height: 40px;
    margin: 8px auto 15px;
    border-bottom: 1px solid #ff6eed;
}

```

- **Ex01.js**

```

/*
기본 Logic
1) 할일(li)을 추가할 대상(ul)선택
2) 할일(li) 목록 생성
3) 파라메타 값으로 전달 받은 인자값(할일) 입력
4) to-do-list에 할일 추가
*/

// 할일(li)을 추가할 대상(ul) 선택
const ToDoList = document.querySelector('#to-do-list')

function newPlan(text){
    // 할일(li) 생성
    const li = document.createElement('li')

    // 생성된 할일(li)에 파라메타로 전달된 인자값 내용(텍스트) 추가
    li.textContent = text

    // '<li>HTML/CSS 공부하기'</li>' 형태로 만들어진 할일 추가
    ToDoList.append(li)
}

newPlan('HTML/CSS 공부하기')
newPlan('JAVASCRIPT 공부하기')
newPlan('DATABASE 공부하기')

```

Ex02. 이메일정보 수집하는 프로그램

- **index.html + Script**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">

```

```

<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Ex06. 이메일정보 수집 프로그램</title>
</head>
<style>
    #submit-btn{
        background: #9b9b9b;
    }
</style>
<body>

    <input id="email-input" type="email" placeholder="이메일">
    <input id="submit-btn" type="submit" value="Send">

<script src="jquery-3.7.0.min.js"></script>
<script>
/*
기본 Logic
1. 이메일에는 @가 포함되어 있어야 한다.
2. @ 뒤에는 최소 1개 이상의 .이 포함되어 있어야 한다
3. 1번과 2번 조건이 모두 충족되었을 때 보내기(send) 버튼 활성화
    1번과 2번 조건 중 하나라도 충족되지 않을 경우 보내기(send) 버튼 비활성화
*/
    $('#email-input').on('input', validateCheck);

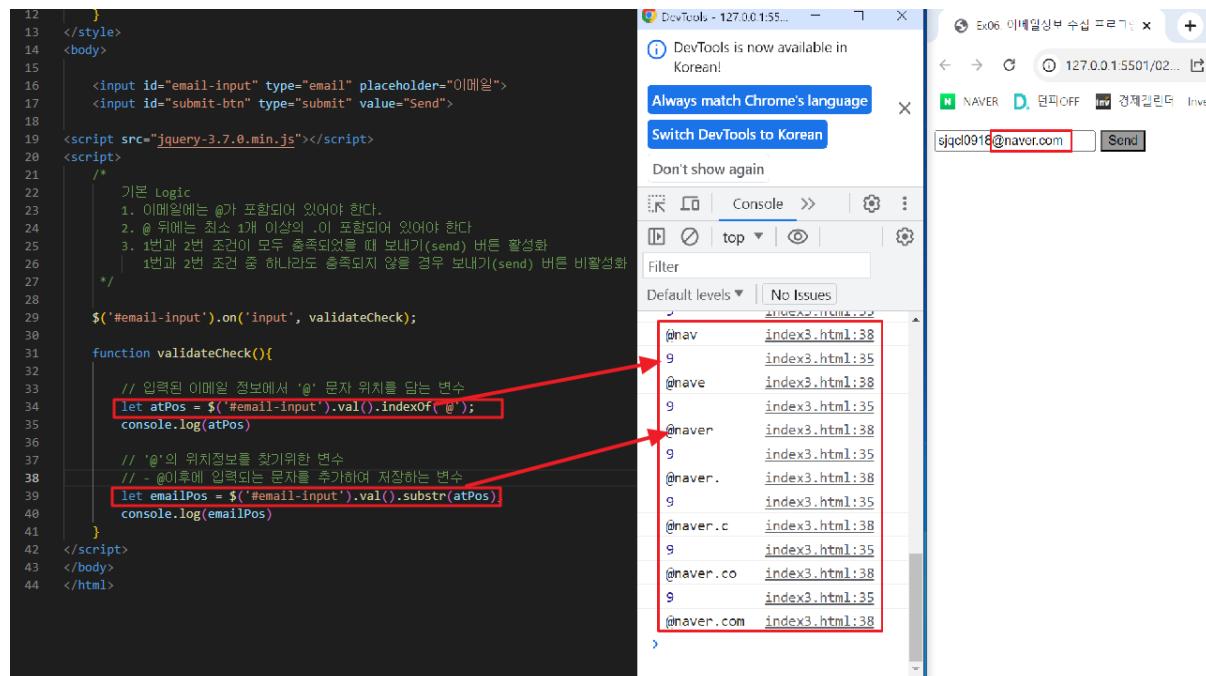
    function validateCheck(){

        // 입력된 이메일 정보에서 '@' 문자 위치를 담는 변수
        let atPos = $('#email-input').val().indexOf('@');
        console.log(atPos)

        // '@'의 위치정보를 찾기위한 변수
        // - @이후에 입력되는 문자를 추가하여 저장하는 변수
        let emailPos = $('#email-input').val().substr(atPos);
        console.log(emailPos)

        // 입력된 이메일에 @와 .이 있다면 '보내기(send) 버튼' 활성화
        if(atPos != -1 && emailPos.indexOf('.') != -1){
            $('#submit-btn').css('background-color', 'blue');
        } else{
            $('#submit-btn').css('background-color', '#9b9b9b');
        }
    }
</script>
</body>
</html>

```



[Project #1 : Up & Down 숫자 맞추기 게임]

[Up & Down Game]

- **Math** : 자바스크립트에서 유용하게 사용되어지는 객체 중 하나. 수학적인 내용의 함수들을 가지고 있다
- **Math.random()** : 0 ~ 1 사이의 값을 반환(1은 미초함)
- **Math.floor()** : 소수점을 버린다
- **Math.ceil()** : 소수점을 올린다
- **Math.round()** : 소수점을 반올림한다
- **Math.max()** : 여러개의 값 중 제일 큰 값을 반환
- **Math.min()** : 여러개의 값 중 제일 작은 값을 반환

• 시작 화면(Cover Page)

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Up & Down Game</title>
    <script src="https://kit.fontawesome.com/0c36059e36.js" crossorigin="anonymous"></script>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Russo+One&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Abrial+Fatface&family=Russo+One&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Abrial+Fatface&family=Jua&family=Russo+One&display=swap" rel="stylesheet">
<style>
    *{
        margin: 0;
        padding: 0;
        box-sizing: border-box;
    }

    a {
        text-decoration: none;
        color: initial;
        text-align: center;
        margin-top: 550px;
    }

    .splash{
        width: 100vw;
        height: 100vh;
        background-color: beige;
        position: absolute;
        top: 0;
        left: 0;
        display: flex;
        justify-content: center;
        align-items: center;
        font-size: 130px;
    }

    .splash span{
        position: relative;
        top: 20px;
        font-family: 'Abrial Fatface', 'cursive';
        animation: bounce .3s ease infinite alternate;
        display: inline-block;
        text-shadow: 0 1px 0 #CCC,
                    0 2px 0 #CCC,
                    0 3px 0 #CCC,
                    0 4px 0 #CCC,
                    0 5px 0 #CCC,
                    0 6px 0 transparent,
                    0 7px 0 transparent,
                    0 8px 0 transparent,
                    0 9px 0 transparent,
                    0 10px 10px rgba(0, 0, 0, .4);
    }

    .splash .fa-comment{
        padding-left: 30px;
        margin-top: 550px;
    }

    .splash span:nth-child(2) {
        animation-delay: .1s;
    }
</style>
```

```

.splash span:nth-child(3) {
    animation-delay: .2s;
}

.splash span:nth-child(4) {
    animation-delay: .3s;
}
.splash span:nth-child(5) {
    animation-delay: .4s;
}
.splash span:nth-child(6) {
    animation-delay: .5s;
}

.splash span:nth-child(7) {
    animation-delay: .6s;
}

.splash span:nth-child(8) {
    animation-delay: .7s;
}

.splash span:nth-child(9) {
    animation-delay: .8s;
}

.tab_hidden , .mo_hidden {
    display: none;
}

@keyframes bounce {
    100% {
        top: -20px;
        text-shadow: 0 1px 0 #CCC,
                    0 2px 0 #CCC,
                    0 3px 0 #CCC,
                    0 4px 0 #CCC,
                    0 5px 0 #CCC,
                    0 6px 0 #CCC,
                    0 7px 0 #CCC,
                    0 8px 0 #CCC,
                    0 9px 0 #CCC,
                    0 50px 25px rgba(0, 0, 0, .2);
    }
}

.svg{
    width: 100%;
    height: 90vh;
    position: absolute;
    bottom: 50px;
    left: 50%;
    transform: translateX(-50%);
    margin-bottom: 100px;
}

.ears{
    animation: moveEars 1s linear infinite alternate;
}

/* 타원의 경우 cx="315" cy="90" 값을 기준점으로 지정해줘야 제자리에서 움직이는 애니메이션 적용 가능 */
.ears:first-child{
    transform-origin: 315px 90px;
    /*
        transform-origin
        => 회전의 중심점(기준점) 지정
        => 기본값 : 50% 50%;
        => 사용단위 : %, 키워드, px
    */

    [백분율과 대응 키워드]

    0% => left, top
    50% => center
    100% => right, bottom
}
.ears:nth-child(2){
    transform-origin: 105px 90px;
}

.cheek{
    animation: cheek 1s linear infinite alternate;
}

```

```

@keyframes moveEars {
    to{
        transform: scale(1.1);
    }
}

@keyframes cheek{
    from{
        transform: translateY(0)
    }
    to{
        transform: translateY(-10px)
    }
}

/* tab */
@media all and (min-width:768px) and (max-width:1023px){

    .fa-comment{
        display: none;
    }
    a{
        font-size: 100px;
    }
}

/* 모바일 */
@media all and (max-width:767px){
    .mo_hidden{
        display: block;
    }
    .fa-comment{
        display: none;
    }
    a{
        font-size: 100px;
        margin-bottom: 50px;
    }
}

```

</style>

</head>

<body>

```

<div class="splash">
    <a href="index.html">
        <svg class="svg" width="420" height="420" viewBox="0 0 420 420" fill="none" xmlns="http://www.w3.org/2000/svg">
            <!-- <rect width="420" height="420" fill="white"/> -->
            <circle class="ears" cx="315" cy="90" r="40" fill="#765049" stroke="black" stroke-width="10"/>
            <circle class="ears" cx="105" cy="90" r="40" fill="#765049" stroke="black" stroke-width="10"/>
            <circle cx="210" cy="210" r="150" fill="#B68278"/>
            <circle cx="210" cy="210" r="150" fill="#B68278"/>
            <circle cx="210" cy="210" r="150" fill="#B68278"/>
            <circle cx="210" cy="210" r="150" stroke="black" stroke-width="10"/>
            <mask id="path-4-inside-1_1_7" fill="white">
                <path fill-rule="evenodd" clip-rule="evenodd" d="M186.693 238H233.522C231.866 228.918 231 219.56 231 210C231 157.07
7.078 257.522 110.351 298 82.3857C273.002 65.1152 242.682 55 210 55C178.793 55 149.74 64.2224 125.418 80.0905C164.531 108.228 190 1
54.14 190 206C190 216.97 188.86 227.674 186.693 238Z" fill="#93665D"/>
            </mask>
            <path fill-rule="evenodd" clip-rule="evenodd" d="M186.693 238H233.522C231.866 228.918 231 219.56 231 210C231 157.07
8 257.522 110.351 298 82.3857C273.002 65.1152 242.682 55 210 55C178.793 55 149.74 64.2224 125.418 80.0905C164.531 108.228 190 1
54.14 190 206C190 216.97 188.86 227.674 186.693 238Z" fill="#93665D"/>
            <path d="M186.693 238L176.906 235.946L174.376 248H186.693V238ZM233.522 238V248H245.512L243.36 236.205L233.522 238ZM
298 82.3857L303.684 90.6131L315.593 82.3857L303.684 74.1584L298 82.3857ZM125.418 80.0905L119.954 71.7153L107.734 79.6877L119.578 8
8.2008L125.418 80.0905ZM186.693 248H233.522V228H186.693V248ZM221 210C221 220.165 221.92 230.124 223.685 239.795L243.36 236.205C241.
811 227.713 241 218.955 241 210H221ZM292.316 74.1584C249.251 103.911 221 153.655 221 210H241C241 160.501 265.793 116.792 303.684 9
0.6131L292.316 74.1584ZM210 65C240.587 65 268.935 74.4598 292.316 90.6131L303.684 74.1584C277.069 55.7766 244.777 45 210 45V65ZM13
0.882 88.4657C153.629 73.6256 180.792 65 210 65V45C176.794 45 145.852 54.8193 119.954 71.7153L130.882 88.4657ZM200 206C200 150.786
172.87 101.908 131.258 71.9728L119.578 88.2082C156.191 114.547 180 157.493 180 206H200ZM196.48 240.054C198.788 229.057 200 217.665
200 206H180C180 216.276 178.933 226.291 176.906 235.946L196.48 240.054Z" fill="black" mask="url(#path-4-inside-1_1_7)"/>
            <circle cx="210.5" cy="253.5" r="22.5" fill="#B68278" stroke="black" stroke-width="10"/>
            <path d="M171 232.5C171 215.655 184.655 202 201.5 202H209V244C209 254.493 200.493 263 190 263C179.507 263 171 254.4
93 171 244V232.5Z" fill="#D9D9D9" stroke="black" stroke-width="10"/>
            <path d="M250 232.5C250 215.655 236.345 202 219.5 202H212V244C212 254.493 220.507 263 231 263C241.493 263 250 254.4
93 250 244V232.5Z" fill="#D9D9D9" stroke="black" stroke-width="10"/>
            <path d="M191.5 200.5H228.5L217 223H204L191.5 200.5Z" fill="#0B0B0B"/>
            <circle class="cheek" cx="200.5" cy="235.5" r="35.5" fill="url(#paint0_linear_1_7)"/>
            <circle class="cheek" cx="130.5" cy="238.5" r="35.5" fill="url(#paint1_linear_1_7)"/>
            <rect x="243" y="177" width="24" height="10" rx="5" fill="black"/>
            <rect x="154" y="177" width="24" height="10" rx="5" fill="black"/>
            <defs>
                <linearGradient id="paint0_linear_1_7" x1="200.5" y1="200" x2="200.5" y2="271" gradientUnits="userSpaceOnUse">
                    <stop stop-color="#FF0000" stop-opacity="0.37"/>
                    <stop offset="1" stop-color="#D9D9D9" stop-opacity="0"/>
                </linearGradient>
                <linearGradient id="paint1_linear_1_7" x1="130.5" y1="203" x2="130.5" y2="274" gradientUnits="userSpaceOnUse">
                    <stop stop-color="#FF0000" stop-opacity="0.37"/>
                    <stop offset="1" stop-color="#D9D9D9" stop-opacity="0"/>
                </linearGradient>
            
```

```

        </linearGradient>
    </defs>
</svg>
<span>G</span>
<span>a</span>
<span>m</span>
<span>e</span>
<br class="tab_hidden mo_hidden">
<span>S</span>
<span>t</span>
<span>a</span>
<span>r</span>
<span>t</span>
</a>
<span>
    <i class="fa-solid fa-comment"></i>
</span>
</div>

</body>
</html>

```

- main page : index.html

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>게임화면</title>
    <link rel="stylesheet" href="style.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Russo+One&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Abril+Fatface&family=Russo+One&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Abrial+Fatface&family=Jua&family=Russo+One&display=swap" rel="stylesheet">
</head>
<body>
    <div class="wrap">
        <div class="in-box">
            <h1>Up & Down 숫자맞추기 게임!</h1>
            <div id="result-area">결과 출력</div>
            <div id="chance-area">남은 칸스 : 7번</div>
            <input id="user-input" type="number" placeholder="1에서 100까지 숫자 중 입력하세요">
            <div class="button_box">
                <button id="play-button">Go!</button>
                <button id="reset-button">Reset!</button>
                <a href="javascript:window.history.back()">
                    <button class="back-button">Back!</button>
                </a>
            </div>
        </div>
    </div>
    <script src="main.js"></script>
    <!-- <div id="text"><span>hello </span> world</div>
    <script>
        let text = document.getElementById('text').textContent
        console.log(text)

        let text2 = document.getElementById('text').innerText
        console.log(text2)

        let text3 = document.getElementById('text').innerHTML
        console.log(text3)
    </script> -->
</body>
</html>

```

- main.js

```

let computerNumber = 0
let chances = 7
let gameOver = false

// 중복된 숫자 입력 방지를 위한 배열 선언
let userValueList = []

```

```

let playButton = document.getElementById('play-button') // id(ById), class(ByClassName), tag(ByTagName)에 따라 다르게 사용된다
// let userInput = document.querySelector('#user-input') // querySelector : 제공한 선택자와 일치하는 요소 반환 (id = # , class = .)
let userInput = document.getElementById('user-input')
let resultArea = document.getElementById('result-area')
let resetButton = document.getElementById('reset-button')
let chanceArea = document.getElementById('chance-area')

// addEventListener('이벤트이름' , 이벤트 발생시 실행시킬 함수)
playButton.addEventListener('click', play)
// 함수의 이름만 전달하는 이유는 click 했을 때 이벤트가 발생하여야 하기 때문이다
// 함수()로 적을 경우 클릭 여부와 상관없이 호출로 인해 바로 실행되어 진다
resetButton.addEventListener('click' , reset)

// 간단한 기능만 구현하는 함수의 경우 익명으로 선언 후 사용 가능
// 1회성으로 사용 시 활용!!
userInput.addEventListener('focus' , function(){
    userInput.value = ""
})

// 난수 생성 함수
function pickRandomNumber(){
    computerNumber = Math.floor(Math.random() * 100) + 1           // Math.random : 0 ~ 1사이의 숫자 랜덤하게 생성 , 1 ~ 100 안의 숫자 지정을 위해
    console.log(computerNumber)
}

// go 버튼 눌렀을 때 이벤트 발생하는 함수
function play(){
    // input에 입력되는 값은 value에 저장된다
    const USER_VALUE = userInput.value      // value : input 태그가 가지고 있는 속성
    console.log(USER_VALUE)
    if(USER_VALUE < computerNumber){
        // console.log("up!!")
        resultArea.textContent = "upup!!"
    }else if(USER_VALUE > computerNumber){
        // console.log("down!!")
        resultArea.textContent = "down!!"
    }else {
        // console.log("딩동댕!!")
        resultArea.textContent = "딩동댕!!"
        // 정답일 경우 go 버튼 비활성화
        gameOver = true
    }

    // 1부터 100사이의 숫자 입력 시 기회가 감소되지 않고 종료
    if(USER_VALUE < 1 || USER_VALUE > 100){
        resultArea.textContent = "1부터 100사이의 숫자를 입력해주세요"
        return
    }
    if(userValueList.includes(USER_VALUE)){
        resultArea.textContent = "이미 입력된 숫자입니다"
        return
    }

    // go버튼이 실행될 때마다 기회가 1회씩 감소
    chances--

    // 입력한 값을 배열에 추가할 때 '입력된 숫자' 여부를 먼저 점검 후 해당 조건이 false 일때 그 값을 배열에 추가하는 로직
    userValueList.push(USER_VALUE)

    // console.log("남은 기회" + chances + "번")
    chanceArea.textContent = `남은 기회 : ${chances}번`
    if(chances == 0){
        gameOver = true
    }
    // gameOver == true라는 것은 위 조건에서 chances == 0이라는 조건이 참이기 때문에
    if(gameOver){
        // disabled : input 태그의 속성
        playButton.disabled = true
    }
}

// rest 버튼 눌렀을 때 이벤트 발생하는 함수
function reset(){
    // reset 버튼을 누르면 난수 초기화
    pickRandomNumber()
    userInput.value = ''
    resultArea.textContent = "결과출력"
    // console.log("리셋")
}

```

- style.css

```

@charset "UTF-8";
@import url("https://fonts.googleapis.com/css2?family=Noto+Sans+KR&display=swap");

*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

button{
    -webkit-appearance: none;
    -moz-appearance: none;
    appearance: none;
}
/* reset */

.wrap{
    width: 100%;
    height: 100vh;
    background: url(img/bg_02.jpg) no-repeat center / cover;
    position: relative;
}

.in-box{
    max-width: 700px;
    width: 90%;
    height: 500px;
    background-color: skyblue;
    border-radius: 20px;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: space-around;
}

.in-box > h1{
    width: 90%;
    height: 70px;
    background-color: beige;
    text-align: center;
    line-height: 70px;
    margin: 20px auto;
    border-radius: 20px;
    font-family: 'Jua', sans-serif;
    font-size: 45px;
}

.in-box > .select-area , #result-area , #chance-area{
    font-size: 35px;
    font-weight: 500;
    font-family: 'Jua', sans-serif;
}

input{
    margin: 0;
    padding: 0.5rem 1rem;
    font-family: "Noto Sans KR", sans-serif;
    font-size: 1rem;
    font-weight: 400;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    width: auto;
    border: none;
    border-radius: 4px;
    box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0, 0.06);
    transition: 0.5s;
}

input{
    width: 300px;
    height: 50px;
    font-family: 'Jua', sans-serif;
    font-size: 18px;
}

button{
    font-family: 'Abril Fatface', 'cursive';
    font-size: 25px;
    background-color: lightgreen;
    border: none;
}

```

```

        cursor: pointer;
        display: inline-block;
        font-size: 22px;
        margin: 15px;
        outline: none;
        padding: 12px 40px 10px;
        position: relative;
        text-transform: uppercase;
        font-weight: 700;
        transition: .5s;
        border-radius: 5px;
    }

    button::before, button::after{
        border-color: transparent;
        -webkit-transition: all 0.25s;
        transition: all 0.25s;
        border-style: solid;
        border-width: 0;
        content: "";
        height: 24px;
        position: absolute;
        width: 24px;
    }

    button::before{
        border-color: lightgreen;
        border-right-width: 3px;
        border-top-width: 3px;
        right: -5px;
        top: -5px;
    }

    button::after{
        border-bottom-width: 3px;
        border-color: lightgreen;
        border-left-width: 3px;
        bottom: -5px;
        left: -5px;
    }

    button:hover , button.hover {
        background-color: lightgreen;
        transform: scale(1.2);
    }

    button:hover::before, button.hover::before, button:hover::after, button.hover::after{
        width: 100%;
        height: 100%;
    }
}

```

※ Up & Down 숫자게임 함수 및 이벤트

- 난수 생성
 - 게임 시작과 동시에 자동으로 1 ~ 100 중 랜덤 숫자 생성

```

function pickRandomNumber(){
    computerNumber = Math.floor(Math.random() * 100) + 1
    console.log(computerNumber)
}
pickRandomNumber()

```
- Go 버튼 눌렀을 때 이벤트
 - 사용자가 입력한 숫자와 컴퓨터가 생성한 난수(정답) 비교 후 조건에 따라 up, down, 딩동댕 출력
 - 사용자가 입력한 숫자가 정답일 경우 Go 버튼 비활성화
 - 1 ~ 100 사이 숫자를 입력하지 않았을 경우 기회가 감소되지 않고 "1에서 100사이 숫자 입력" 출력 후 종료
 - 사용자가 입력한 숫자를 순차적으로 배열에 저장하여 충복 입력 방지
 - 숫자 1회 입력할 때마다 찬스 1회씩 감소
 - 찬스가 0일 경우 게임 종료 및 Go 버튼 비활성화

```

function play(){

    const USER_VALUE = userInput.value
    console.log(USER_VALUE)
    if(USER_VALUE < computerNumber){
        resultArea.textContent = "upup!!"
    }else if(USER_VALUE > computerNumber){
        resultArea.textContent = "down!!"
    }else {
        resultArea.textContent = "딩동댕!!"
        gameOver = true
    }

    if(USER_VALUE < 1 || USER_VALUE > 100){

```

```

        resultArea.textContent = "1부터 100사이의 숫자를 입력해주세요"
        return
    }
    if(userValueList.includes(USER_VALUE)){
        resultArea.textContent ="이미 입력된 숫자입니다"
        return
    }
    chances--
    userValueList.push(USER_VALUE)

    chanceArea.textContent = `남은 기회 : ${chances}번`
    if(chances == 0){
        gameOver = true
    }
    if(gameOver){
        // disabled : input 태그의 속성
        playButton.disabled = true
    }
}

3. 숫자 입력 후 Enter key를 눌렀을 때 Go 버튼을 누른것과 동일한 효과 주기
- 키보드 Enter Key 입력 시 입력창 내용 삭제
userInput.addEventListener('keyup', function(){
    if(window.event.keyCode == 13){
        play()
        userInput.value =''
    }
})

4. 숫자 입력 후 입력창에 마우스 커서 펑업 시
userInput.addEventListener('focus' , function(){
    userInput.value = ""
})

5. reset 버튼 눌렀을 때 이벤트
- 난수 초기화
- 사용자가 입력창에 입력한 숫자 삭제
- 남은 칸스 7회로 초기화
function reset(){
    pickRandomNumber()
    userInput.value = ''
    resultArea.textContent = "결과출력"
    chances = 7
    chanceArea.textContent = `남은 기회 : ${chances}번`
}

```

[Project #2 : ToDoList 만들기]

1) 참고 : <https://non-stop.tistory.com/400>

- index.html #1

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ToDoList</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Abribl+Fatface&family=Bruno+Ace&family=Jua&family=Russo+One&display=swap" rel="stylesheet">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2VBKQH2PbZ58rQ0国家安全法">
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="wrap">
        <div class="in-box">
            <div class="left_wrap">
                <div class="bgimg_box">
                    <div class="bg_img"></div>
                    <p class="name">Do It<br class="mo_hidden tab_hidden">Bae Sang Won</p>
                    <p class="tab_name">Bae Sang Won</p>
                </div>
                <div class="task_box">
                    <p class="today">Month Task</p>
                    <div>
                        <input style="zoom: 2.0;" class="check" type="checkbox" id="work1">
                        <label for="work1"><span>WORK 01</span></label>
                        <button>삭제</button>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

<div>
    <input style="zoom: 2.0" class="check" type="checkbox" id="work2">
    <label for="work2"><span>WORK 02</span></label>
    <button>삭제</button>
</div>
<div>
    <input style="zoom: 2.0" class="check" type="checkbox" id="work3">
    <label for="work3"><span>WORK 03</span></label>
    <button>삭제</button>
</div>
</div>
<div class="right_wrap">
    <h1>To Do List</h1>
    <input id="todo" type="text" placeholder="할일을 입력하세요">
    <button id="addButton">추가</button>
    <div id="ToDoList">
        <div>
            <input type="checkbox" id="check-box">
            <label for="check-box"><span>example</span></label>
            <button>삭제</button>
        </div>
    </div>
</div>
<script src="main.js"></script>
</body>
</html>

```

- **main.js #1**

```

// DOMContentLoaded : HTML문서를 완전히 불러오고 분석했을 때 이벤트 발생
/*
 * 람다식 함수
 - 코드간략화를 위해 사용
 - 선언한 시점에 this를 확보하기 때문에, 한번 확보된 값은 고정으로 설정되어 변경되지 않는다
 즉, 다른 곳에서 함수를 호출하여도 처음에 호출되었을 때 설정된 값이 계속 출력된다
*/
document.addEventListener('DOMContentLoaded', () => {
    const todo = document.querySelector('#todo')
    const addButton = document.querySelector('#addButton')
    const ToDoList = document.querySelector('#ToDoList')

    addButton.addEventListener('click', (event) => {
        // createElement : 지정한 태그내입(div) 생성
        const item = document.createElement('div')
        const checkBox = document.createElement('input')
        // element.setAttribute( 'attributename(속성이름)', 'attributevalue(속성값)' )
        checkBox.setAttribute('type', 'checkbox')

        const text = document.createElement('span')
        text.textContent = todo.value

        const removeButton = document.createElement('button')
        removeButton.textContent = '삭제'

        removeButton.addEventListener('click', (event) => {
            // currentTarget : 이벤트 핸들러가 부착된 것을 가리킨다
            // - event.target 는 부모로부터 이벤트가 위임되어 발생하는 자식의 위치, 내가 클릭한 자식 요소 반환
            // - event.currentTarget 는 이벤트가 부착된 부모의 위치 반환
            // parentNode : 부모노드, childNodes : 자식노드 리스트, nextSibling : 다음 형제노드, previousSibling : 이전 형제노드
            // removeChild : 부모에서 포함된 자식 노드가 존재할 경우 일치하는 ID, Class 등과 같은 속성을 통해 자식 노드를 제거
            event.currentTarget.parentNode.removeChild(event.currentTarget.parentNode)
        })

        checkBox.addEventListener('change', (event) => {
            if(checkBox.checked){
                // 체크박스에 체크될 경우 텍스트 중간에 선을 만든다
                text.style.textDecorationLine = "line-through"
            }else{
                text.style.textDecorationLine = "none"
            }
        })
        // append 메서드 : 노드객체(Node object)나 DOMString(text) 사용 가능
        // append 메서드는 return 값 반환 x
        // appendChild 메서드는 오직 Node 객체만 받을 수 있으며 한번에 하나의 노드만 추가 가능
        /*
         *append 와 appendChild 사용 예시
         - append : item.append(div, span, p)
         - appendChild : item.appendChild(text)
        */
    })
})

```

```

        item.appendChild(checkBox)
        item.appendChild(text)
        item.appendChild(removeButton)
        todoList.appendChild(item)
        todo.value = ''
    })
})

// 키보드 Enter Key를 눌렀을 때 add버튼 클릭과 동일한 이벤트 적용
// click() 함수 : 마우스 왼쪽버튼으로 선택 요소를 클릭하면 발생
// *addButton의 이벤트 발생 조건이 'click'이기 때문에 입력창에 내용 입력 후 Enter를 입력하면 addButton을 마우스 왼쪽버튼으로 클릭 실행
// 즉, Enter key 입력 = addButton.addEventListener('click', event)와 같은 내용
todo.addEventListener('keyup', function(){
    if(window.event.keyCode == 13){
        addButton.click()
    }
})

```

- **style.css #1**

```

@charset "utf-8";

*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

.wrap{
    width: 100%;
    height: 100vh;
    background-color: #C5a4fa;
    position: relative;
}

.in-box{
    max-width: 900px;
    width: 90%;
    height: 700px;
    border-radius: 20px;
    position: absolute;
    top: 50%;
    left: 56%;
    transform: translate(-50%, -50%);
    display: flex;
    border: 3px solid #d070fb;
}

.left_wrap{
    width: 100%;
    background-color: #fff;
    border-radius: 20px 0 0 20px;
}

.bgimg-box{
    width: 100%;
    height: 150px;
    display: flex;
    justify-content: space-around;
    padding: 20px;
}

.bg_img{
    width: 100px;
    height: 100px;
    border-radius: 50%;
    border: 1px solid gray;
    background: url('img/mimoticon_4.png') no-repeat center / cover;
}

.name{
    width: calc(100% - 150px);
    height: 100px;
    font-size: 30px;
    text-align: center;
    line-height: 50px;
    font-weight: bold;
    font-family: 'Bruno Ace', 'cursive';
}

.tab_name{

```

```

width: calc(100% - 70px);
height: 100px;
text-align: center;
line-height: 100px;
display: none;
font-family: 'Bruno Ace', 'cursive';
font-size: 23px;
}

.task-box{
    width: 100%;
    height: calc(100% - 200px);
}

.task-box::before{
    content: "";
    display: block;
    width: 80%;
    border-bottom: 3px solid #C5a4fa;
    margin: 0 auto;
}

.today{
    width: 100%;
    height: 50px;
    font-size: 25px;
    margin-top: 20px;
    font-weight: bold;
    font-family: 'Bruno Ace', 'cursive';
    text-align: center;
    line-height: 50px;
}

.task-box div{
    width: 80%;
    height: 50px;
    display: flex;
    justify-content: space-around;
    align-items: center;
    background-color: rgba(208, 112, 251, .7);
    color: white;
    border-radius: 20px;
    margin: 10px auto;
    line-height: 50px;
    font-size: 20px;
    font-family: 'Bruno Ace', 'cursive';
}

.task-box div button{
    width: 50px;
    height: 35px;
    font-size: 17px;
    font-family: 'Jua', sans-serif;
    border-radius: 5px;
    border: none;
    cursor: pointer;
}

.right_wrap{
    max-width: 700px;
    width: 90%;
    height: 100%;
    background-color: blueviolet;
    border-radius: 0 20px 20px 0px;
}

.right_wrap > h1{
    width: 90%;
    height: 80px;
    color: #fff;
    font-family: 'Bruno Ace', 'cursive';
    font-size: 50px;
    text-align: center;
    line-height: 80px;
    margin: 0 auto;
}

#ToDo, #addButton{
    width: 80%;
    height: 50px;
    font-family: 'Jua', sans-serif;
    font-size: 20px;
    font-weight: 400;
    text-align: center;
    text-decoration: none;
    display: inline-block;
}

```

```

        border: none;
        border-radius: 5px;
        box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0, 0.06);
        transition: 0.5s;
        margin-left: 20px;
        cursor: pointer;
    }

    #addButton{
        width: 50px;
        margin: 0;
    }

    #ToDoList{
        width: 100%;
        height: calc(100% - 150px);
    }

    #ToDoList > div{
        width: 90%;
        height: 40px;
        font-family: 'Bruno Ace', 'cursive' , 'Jua', sans-serif;
        font-weight: 500;
        font-size: 25px;
        text-decoration: none;
        margin: 20px auto;
        text-transform: uppercase;
        color: #fff;
        display: flex;
        justify-content: space-between;
    }

    #ToDoList > div > #check-box{
        padding: 10px;
    }

    #ToDoList > div > input[type=checkbox]{
        transform: scale(2);
    }

    #ToDoList > div > span{
        border-bottom: 3px solid salmon;
    }

    #ToDoList > div > button{
        width: 50px;
        font-size: 17px;
        font-family: 'Jua', sans-serif;
        text-align: center;
        border-radius: 5px;
        border: none;
        cursor: pointer;
    }

    #ToDoList > div > input, span, button{
        margin-left: 10px;
    }

    /* tab */
    @media all and (min-width:768px) and (max-width:1023px){
        .name{
            display: none;
        }

        .tab_name{
            display: block;
        }

        .right_wrap > h1{
            font-size: 35px;
        }

        #ToDo{
            width: 200px;
        }
    }

    /* 모바일 */
    @media all and (max-width:767px){
        .name{
            display: none;
        }

        .task-box div{
    
```

```

        width: 90%;
    }

    .task-box div span{
        font-size: 13px;
    }

    .right_wrap{
        width: 500px;
    }

    .right_wrap > h1{
        font-size: 30px;
    }

    #ToDo{
        width: 130px;
    }
    #ToDo::placeholder {
        font-size: 15px;
    }

    #ToDoList > div{
        height: 30px;
        display: flex;
        justify-content: space-between
    }

    input[type=checkbox]{
        transform: scale(.7);
    }

    #ToDoList > div > span{
        font-size: 18px;
        margin: 0;
    }

    #ToDoList > div > input, span, button{
        margin-left: 3px;
    }

    #ToDoList > div > button, .task-box div button{
        width: 30px;
        font-size: 12px;
    }
}

```

※ JavaScript 일반 함수와 람다 함수 비교

1) 일반함수

일반 함수 this 예제

```

param = 'global param';

function printParam(){
    console.log(this.param);
}

let object = {
    param: 'object param',
    func: printParam
}
let object2 = {
    param: 'object2 param',
    func: printParam
}

object.func();
object2.func();

```

결과

```

object param
object2 param

```

람다식 함수 this 예제

```

param = 'global param';

let printParam = () => {
    console.log(this.param);
}

let object = {
    param: 'object param',
    func: printParam
}
let object2 = {
    param: 'object2 param',
    func: printParam
}

object.func();
object2.func();

```

결과

```

global param
global param

```

2) 람다식 함수

- 코드간략화를 위해 사용

- 선언한 시점에 `this`를 확보하기 때문에, 한번 확보된 값은 고정으로 설정되어 변경되지 않는다
- 즉, 다른 곳에서 함수를 호출하여도 처음에 호출되었을 때 설정된 값이 계속 출력된다
- 아래의 코드에서처럼 `JavaScript`에서 `var`나 `let`을 사용하지 않고 선언한 변수는 전역 변수됨
- 함수를 호출한 오브젝트가 존재하지 않는 상태에서는 `this`는 전역변수를 가르킴
- `printParam` 함수 안의 `this`도 전역변수를 가르키게됨

※ `target` 과 `currentTarget` 비교

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>target 과 currentTarget 비교</title>
</head>
<body>
  <li>
    <button onClick={onLogin}>
      <span>Button</span>
    </button>
  </li>

  <script src="taget.js"></script>
</body>
</html>
```

```
const onLogin = (event) => {
  console.log(event.target);
  console.log(event.currentTarget);
};
```

<code>Google</code>	<u>login.jsx:8</u>
▼ <code><button></code>	<u>login.jsx:9</u>
<code> Google</code>	
<code> </button></code>	

정리

`event.target`은 자식 요소인 `span`을 리턴하고, `event.currentTarget`은 부모 요소인 `button`을 반환하는 것을 알 수 있다.

※ `removeChild`

- 부모에서 포함된 자식 노드가 존재할 경우 일치하는 ID, Class 등과 같은 속성을 통해 자식 노드를 제거

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>removeChild</title>
</head>
<body>
  <div id="par_div">
    <div id="child_div1">첫번째 div</div>
    <div id="child_div2">두번째 div</div>
    <div id="child_div3">세번째 div</div>
  </div>
</body>
</html>
```

```

var parent = document.getElementById("par_div");
var child = document.getElementById("child_div2");

parent.removeChild(child);

if(child.parentNode){
    child.parentNode.removeChild(child);
}

var child = document.getElementById("child_div4");
parent.removeChild(child4);
parent.removeChild(child);

```

```

1  <!DOCTYPE html>
2  <html lang="ko">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>removeChild</title>
8  </head>
9  <body>
10     <div id="par_div">
11         <div id="child_div1">첫번째 div</div>
12         <div id="child_div2">두번째 div</div>
13         <div id="child_div3">세번째 div</div>
14     </div>
15
16     <script>
17         var parent = document.getElementById("par_div");
18         var child = document.getElementById("child_div2");
19
20         parent.removeChild(child);
21
22         if(child.parentNode){
23             child.parentNode.removeChild(child);
24         }
25
26         var child = document.getElementById("child_div4");
27         parent.removeChild(child4);
28         parent.removeChild(child);
29     </script>
30 </body>
31 </html>

```



- **click()** 함수 : 마우스 왼쪽버튼으로 선택 요소를 클릭하면 발생

2) SBS 아카데미 - 웹컴바인3 (with. 허지훈 강사)

- **index.html #2**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>todolist</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Abrial+Fatface&family=Bruno+Ace&family=Jua&family=Russo+One&display=swap" rel="stylesheet">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2VBKQhggWzH7PCCaAq046Mgn0M80zW1RwUH61DGLwZJEdK2Kadq2F9CUG65" crossorigin="anonymous">
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- container : bootstrap 에서 제공하는 class name -->
    <div class="container">
        <h1>To Do List</h1>
        <section class="input-area">
            <input type="text" id="task-input" placeholder="할일을 입력하세요">
            <button id="add-button">
                <i class="fa-solid fa-square-plus"></i>
            </button>
        </section>
        <section class="task-area">
            <div class="task-tabs">
                <!-- js에서 가져와 사용하기 위해 ID로 부여 -->
                <div id="under-line"></div>
                <div id="all">모두</div>
                <div id="ongoing">진행중</div>

```

```

        <div id="done">완료</div>
    </div>
    <!-- 할일을 추가할 때 task class 내부 전체 요소가 같이 추가되도록 하기 -->
    <div id="task-board">
        <!-- <div class="task">
            <div>치과가기</div>
            <div>
                <button>체크</button>
                <button>삭제</button>
            </div>
        </div> -->
    </div>

    </section>
</div>

<script src="https://kit.fontawesome.com/8c36059e36.js" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBle4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4" crossorigin="anonymous"></script>
<script src="main.js"></script>
</body>
</html>

```

- **main.js #2**

```

/*
[기본 Logic]
1. 유저가 값을 input에 입력한다.
2. + 버튼을 클릭하면 아이템이 더해진다. 할일이 추가된다.
3. 삭제 버튼을 클릭하면 할일이 삭제된다.
4. 체크 버튼을 누르면 할일이 끝나면서, 줄이 그어진다
5. 진행 중, 완료 템을 누르면 인더바가 이동한다.
6. 완료 템에는 완료 아이템만, 진행 중 템에는 진행중인 아이템만 보여진다. 모두 템을 누르면 전체 아이템을 보여준다
*/

```

```

// taskList : 입력된 할일 목록 저장용 변수
let taskList = []

// mode : event.target.id 값을 저장용 변수
// mode의 초기값이 빈 배열로 선언된 경우 초기에 할일 입력 시 입력된 내용이 출력되지 않고, '모두(mode = 'all')'를 클릭하여야 내용이 보여진다 (하단 if문 참(true) 조건)
// 따라서 입력된 할일 리스트를 바로 출력해주기 위해 mode 초기값을 'all' 상태로 저장한다
let mode = 'all'

// filterList : 체크박스에 체크가 되지 않은 할일 항목 저장용 변수
let filterList = []

let taskInput = document.getElementById('task-input')
let addButton = document.getElementById('add-button')
let tabs = document.querySelectorAll('.task-tabs div')
let underline = document.getElementById('under-line')

addButton.addEventListener('click', addTask)

// 입력창에 마우스 커서 팝업 시 내용 삭제
taskInput.addEventListener('focus', function(){
    taskInput.value = ""
})

// 키보드 Enter key 입력 시 추가 버튼 눌른 것과 동일한 효과 적용
taskInput.addEventListener('keyup', function(){
    if(window.event.keyCode == 13){
        addTask()
    }
})

// 모두, 진행중, 완료 tab 기능 구현
for(let i = 1 ; i < tabs.length; i++){
    tabs[i].addEventListener('click', function(event){filter(event)})
}

// 할일 추가 함수
// 1) input에 입력한 값을 가지고와 taskList 변수에 배열로 저장
function addTask(){
    // task 객체에 input에 입력된 내용 + 진행중 or 완료 여부까지 포함된 값 저장
    let task = {
        id: randomIDGenerate(),           // 입력된 할일에 각각 유일한 ID값을 부여
        taskContent: taskInput.value,      // input에 입력된 내용을 저장하는 요소
        isComplete : false                // 진행중 or 완료여부를 판단할 수 있는 요소
    }
    taskList.push(task)
    render()
}

```

```

}

// 추가한 할일 출력 함수
// 1) for문 : taskList 배열에 길이값을 순차적으로 비교
// 2) if문 : 할일이 완료 상태라면 task-done을 호출하여 가운데 줄을 굽는 상태로 만든다
// 3) task-board의 전체 요소(태그포함)를 resultHTML에 저장
// 4) 체크 버튼 클릭 시 할일의 체크 여부 상태에 따라 "진행 중" "완료" 상태를 구분하는 이벤트 실행
// 5) list 배열에 할일 상태가 '모두'일 경우 할일리스트(taskList) 전부를 '진행 중' 상태일 경우 filterList 정보를 저장 및 출력
function render(){
    let list = [] // taskList(할일) 배열에 입력된 내용을 mode('모두', '진행중', '완료') 상태에 따라 list 배열에 저장 및 출력
    if(mode == 'all'){
        list = taskList
    } else if(mode == 'ongoing' || mode == 'done'){ // mode = 'all' 인 상태를 제외하고 filterList 정보 출력
        list = filterList
    }
    let resultHTML = ''
    for(let i = 0; i < list.length; i++){
        if(taskList[i].isComplete == true){ // task-done class에 가운데 줄을 굽는 css 반영
            resultHTML += `
<div class="task">
    <div class="task-done">${list[i].taskContent}</div>
    <div>
        <button onclick="toggleComplete('${list[i].id}')">체크</button>
        <button onclick="deleteTask('${list[i].id}')">삭제</button>
    </div>
</div>`
        } else { // task 내부 요소 전부를 resultHTML에 누적시켜서 저장
            // - taskList 배열에 있는 객체 요소중 입력된 내용 요소만 resultHTML에 저장
            resultHTML += `
<div class="task">
    <div>${list[i].taskContent}</div>
    <div>
        <button onclick="toggleComplete('${list[i].id}')">체크</button>
        <button onclick="deleteTask('${list[i].id}')">삭제</button>
    </div>
</div>`
        }
        // button onclick : addEventListener을 대신해 HTML요소에 바로 동적 이벤트 적용 -> onclick="toggleComplete()"
    }
    // ${taskList[i].taskContent} : taskList 배열에 저장된 값중 내용 요소 , '${taskList[i].id} : taskList 배열에 저장된 값중 내용의 고유ID 요소
    document.getElementById('task-board').innerHTML = resultHTML
    // 추가 버튼을 눌렀을 경우에 할일 입력창의 내용 삭제
    taskInput.value = ""
}

// 체크버튼 클릭시 기능 실행 함수
// 1) 할일에 체크를 하면 완료상태로 변경
function toggleComplete(id){
    // taskList 배열의 값의 id가 전달받은 id와 같다면 해당 할일을 완료 상태로 변경
    for(let i = 0 ; i < taskList.length ; i++){
        if(taskList[i].id == id){
            // 체크박스 체크 해제시 '완료' 상태를 '진행중' 상태로 변경
            // not 연산자를 이용하여 결과를 반대로 바꾼다
            taskList[i].isComplete = !taskList[i].isComplete
            // 체크 여부에 따라 '진행중' 탭과 '완료' 탭에 있는 항목을 각각 이동(삭제) 한다
            // ex) '진행중' 탭에서 항목에 체크할 경우 체크된 항목이 삭제되며, '완료' 탭에서 항목이 보여짐과 동시에 체크버튼 -> 체크해제 버튼으로 바뀜
            filterList.splice(i,1)
            break
        }
    }
    //rander()
    filter() // filter()함수를 통해서 render()가 실행되도록
}

// 삭제버튼 클릭시 기능 실행 함수
// 1) 할일 삭제 버튼 누르면 삭제
// 2) 완료 탭에서 삭제 버튼 누르면 바로 삭제되도록 변경
function deleteTask(id){
    // taskList 배열의 값의 id가 전달받은 id와 같다면 삭제를 누른 배열 요소 1개 삭제
    for(let i = 0 ; i < taskList.length ; i++){
        if(taskList[i].id == id){
            taskList.splice(i, 1)
            break
        }
    }
    //rander()
    filter()
    // filterList 배열의 값과 id가 전달받은 id와 같다면 삭제를 누른 배열 요소 1개 삭제
}

// 모두, 진행중, 완료 tab 기능 실행 함수
// filter : 선택한 요소의 좌표값, 위치정보 등 모든 정보를 보여준다
// 1) if, for문 : mode : 'all', 'ongoing', 'done'에 따라 할일 리스트를 다르게 보여준다
// 2) tab을 이동할 때 마다 밀줄 속성이 따라온다 (현재 선택한 탭이 어떤 것인지를 표시해주기 위함)
function filter(event){
}

```

```

filterList = []
// if문 : event 값이 존재하면, 즉 최초 '모두' 상태에서 '진행중' 상태로 넘어갈 경우 event값이 발생(true)함에 따라 mode = id값을 가져와 if절 실행
// event : tabs가 실행될 경우에 발생
if(event){
    // mode : 'all', 'ongoing', 'done' 을 저장
    mode = event.target.id
    // 'under-line(밑줄)' 의 target 속성 중 너비, 높이 좌표 값 속성을 가지고 와서 탭의 각 항목을 누를때마다 따라가도록 설정
    // .style.width는 css에 반영해준 값을 그대로 적용한다
    // offsetWidth : margin을 제외한 padding, border 값까지 계산한 너비 값을 가져온다
    underline.style.width = event.target.offsetWidth + 'px' // under-line의 width값을 underline이 가지는 영역만큼으로 바꾸겠다는 속성
    underline.style.top = '58px'
    underline.style.left = event.target.offsetLeft + 'px' // offsetLeft : 해당요소의 위치값 만큼 이동하는 속성
}
if(mode == 'all'){
    // 'all', 'ongoing' , 'done' 상태에 따라 밑줄 및 글자 색상 변경
    // all : 밑줄(red), 글자색(white) , ongoing : 밑줄(yellow), 글자색(white)..
    underline.style.backgroundColor = 'red'
    document.getElementById('all').style.color = 'white'
    document.getElementById('ongoing').style.color = 'black'
    document.getElementById('done').style.color = 'black'
} else if(mode == 'ongoing'){
    underline.style.backgroundColor = 'yellow'
    document.getElementById('all').style.color = 'black'
    document.getElementById('ongoing').style.color = 'white'
    document.getElementById('done').style.color = 'black'
    for(let i = 0 ; i < taskList.length; i++){
        if(taskList[i].isComplete == false){ // 할일 리스트 중 '진행중' 상태인 요소만 리스트 배열에 추가
            filterList.push(taskList[i])
        }
    }
} else if(mode == 'done'){
    underline.style.backgroundColor = 'blue'
    document.getElementById('all').style.color = 'black'
    document.getElementById('ongoing').style.color = 'black'
    document.getElementById('done').style.color = 'white'
    for(let i = 0; i < taskList.length; i++){
        if(taskList[i].isComplete == true){ // 할일 리스트 중 '완료' 상태인 요소만 리스트 배열에 추가
            filterList.push(taskList[i])
        }
    }
    // '덮어쓰기' : filter 함수 내에서 지역변수로 선언된 filterList의 값을 전역변수에 덮어쓰기를 할 경우
    // 입력된 할일 리스트 값을 빈 배열인 filterList = [] 값으로 덮어쓰기 때문에 filterList는 전역변수로 선언해준다
    // taskList = filterList --> 이 경우 원래 담고 있는 할일 리스트의 값이 삭제됨(버전관리 x)
}
render()
}

// 유일한 ID 값을 반환해주는 함수
function randomIDGenerate(){
    return Math.random().toString(36).substr(2, 16);
}

5/2 과제 풀이
1. UnderLine 꾸미기

2. 체크 / 삭제버튼 꾸미기
=> css 적용

★3. 체크 버튼 눌렀을 때 refresh 버튼으로 바꾸기
=> '진행중' 탭에 있는 항목중 체크버튼 누른 항목 바로 '완료' 탭으로 이동되는 것처럼 보여지게
=> toggleComplete 함수 for문에 filterList.splice(i,1) 추가

4. Task 배경색 바꿔도록

5. Enter key 눌렀을 때 입력되도록
=> 1) taskInput.addEventListener('keyup', function(){
    if(window.event.keyCode == 13){
        addButton.click()
        taskInput.value = ""
    }
})
=> 2) taskInput.addEventListener('keyup', function(){
    if(window.event.keyCode == 13){
        addTask()
    }
})

6. input 창 클릭하면 input 창 내용 없애기 or Enter 쳤을 때 바로 input 창 내용 없애기
=> 1) taskInput.addEventListener('focus', function(){
    taskInput.value = ""
})
=> 2) render 함수 마지막 구문에
taskInput.value = "" 를 추가하면 추가버튼을 누름과 동시에 내용 삭제됨
=> 3) function addTask(){


```

```

        let task = {
            id: randomIDGenerate(),
            taskContent: taskInput.value,
            isComplete : false
        }
        taskList.push(task)
        taskInput.value = ""
        render()
    }

7. 삭제 버튼 누르면 바로 삭제 되도록
=> 1) deleteTask 함수에 for문 추가
filterList.length 값이 전달받은 id와 같으면 삭제 버튼을 누른 filterList 요소 1개 삭제

for(let i = 0 ; i < filterList.length ; i++){
    if(filterList[i].id == id){
        filterList.splice(i, 1)
    }
}

2-1) 함수 호출 구문 수정
- deleteTask, toggleComplete 함수에서 render() 호출을 filter() 호출로 변경
※ filter() 함수에서도 render() 함수를 호출하기 때문에 바로 render() 함수를 호출하는 것 대신
filter() 함수 호출 후 filter() 함수에서 render() 함수를 실행하도록 수정

2-2) if문 추가
※ default 값으로 mode가 'all' 상태이다
최초 실행시 mode 가 'all' 인 상태이기 때문에 event.target.id 정보를 가져올 수 없기 때문에
'모두' 상태에서 '진행중' 상태로 넘어갈 경우(탭항목을 클릭했을 때 발생) event값이 발생(true)함에 따라 mode = id값을 가져와 if문 실행

```

• style.css #2

```

@charset "UTF-8";
@import url("https://fonts.googleapis.com/css?family=Poppins:200,300,400,500,600,700,800,900&display=swap");

*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body{
    background: url(img/bg02.jpg) no-repeat center / cover;
}

.container{
    min-height: 100vh;
    background-color: rgba(0, 0, 0, .2);
    box-shadow: 10px 5px 5px gray;
}

.container > h1{
    font-family: 'Abril Fatface', 'cursive';
    font-size: 60px;
}

.input-area{
    width: 100%;
    height: 100px;
    display: flex;
    align-items: center;
}

.input-area > #task-input{
    width: 300px;
    height: 50px;
    font-size: 25px;
    font-weight: 400;
    font-family: 'Jua', sans-serif;
    text-decoration: none;
    text-align: center;
    border: none;
    box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0, 0.06);
    border-radius: 5px;
    cursor: pointer;
}

/* input box에 커서를 위치하면 생기는 테두리 삭제 */
#task-input:focus{
    outline: none;
}

```

```

#add-button{
    width: 50px;
    /* height 값을 0으로 주면 버튼의 기능은 적용되면서 아이콘 요소 뒤에 가릴 수 있음 */
    height: 50px;
    border: none;
    margin-left: 10px;
    border-radius: 5px;
    background-color: rgba(0, 0, 0, .1);
}

#add-button > .fa-square-plus{
    font-size: 50px;
    color : #fff
}

.task-area{
    border: 2px solid lightslategray;
}

.task-tabs{
    display: flex;
    border-bottom: 2px solid lightgrey;
    position: relative;
}

.task-tabs div{
    padding: 1em;
    cursor: pointer;
}

.under-line{
    width: 80px;
    height: 4px;
    padding: 0;
    position: absolute;
    left: 0;
    top: 58px;
    background-color: salmon;
    transition: all .3s;
}

.all, #ongoing, #done{
    font-size: 18px;
    font-weight: 600;
    font-family: 'Jua', sans-serif;
}

.bi-arrow-counterclockwise, .bi-trash3, .bi-check{
    width: 30px;
    height: 30px;
}

.task{
    width: 100%;
    height: 100%;
    /* background-color: red; */
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1em;
}

.task-done, .task-done-2{
    width: calc(100% - 230px);
    height: 50px;
    padding: 10px;
    font-size: 25px;
    font-weight: 400;
    font-family: 'Jua', sans-serif;
    text-decoration: none;
    line-height: 40px;
    background-color: rgba(0, 0, 0, 0.1);
    border: none;
    box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0, 0.06);
    border-radius: 5px;
    cursor: pointer;
}

.task > div > button{
    padding: 5px 20px;
    border: 3px solid lightskyblue;
    background-color: lightskyblue;
    /* color: #6e6e6e; */
    color: #fff;
    font-size: 20px;
    border-radius: 15px;
}

```

```

        font-family: 'Jua', sans-serif;
        box-shadow: 0 15px 35px rgba(0, 0, 0, 0.2);
        text-decoration: none;
        transition: 0.25s;
    }

    .task-done {
        text-decoration: line-through;
    }

```

※ JavaScript 유일한 ID 만들기

```

let newID = function () {
    // toString(36) : 36진수, 1~Z까지
    return Math.random().toString(36).substr(2, 16);
}
console.log(newID());

```

```

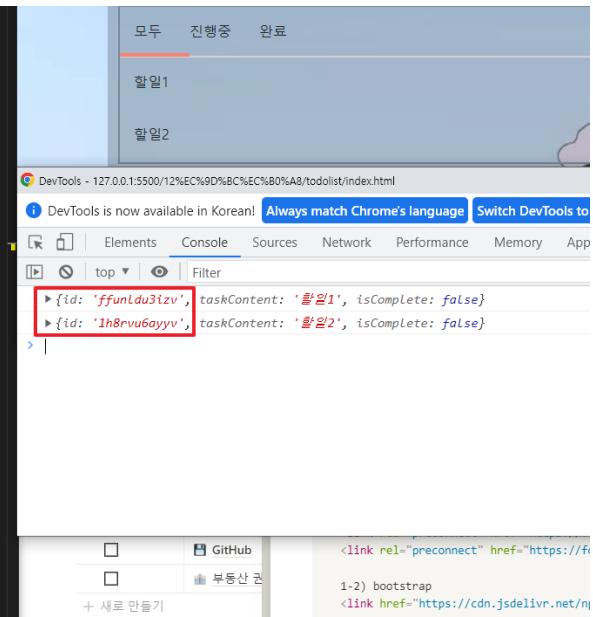
        taskContent: taskInput.value,           // input에 입력된 내용을 저장하는 요소
        isComplete : false                   // 진행중 or 완료여부를 판단할 수 있는
    }
    taskList.push(task)
    console.log(task)
    render()
}

// 추가한 할일 출력 함수
// 1. task-board의 전체 요소(태그포함)를 resultHTML에 저장
function render(){
    let resultHTML = ''
    // taskList 배열에 길이값을 순차적으로 비교
    for(let i = 0; i < taskList.length; i++){
        // task 내부 요소 전부를 resultHTML에 누적시켜서 저장
        // - taskList 배열에 있는 객체 요소중 입력된 내용 요소만 resultHTML에 저장
        // addEventListener를 대신해 HTML요소에 바로 풍적 이벤트 적용 -> onclick=""
        resultHTML += `
            <div class="task">
                <div>${taskList[i].taskContent}</div>
                <div>
                    <button onclick="toggleComplete('${taskList[i].id}')">체크</button>
                    <button>삭제</button>
                </div>
            </div>
        `
    }
    document.getElementById('task-board').innerHTML = resultHTML
}

// 체크버튼 클릭시 기능 실행 함수
function toggleComplete(){
    console.log("체크")
}

// 유일한 ID 사용 함수
function randomIDGenerate(){
    return Math.random().toString(36).substr(2, 16);
}

```



※ google font & bootstrap HTML , JS Link

```

1. HTML

1-1) google font
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

1-2) bootstrap
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2VbkQhgwwz

2. JavaScript

2-1) bootstrap
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBIE4zVF0s0G1M5b4h

```

- **offsetWidth**

```

document.getElementById('under-line').style.width = event.target.offsetWidth + 'px' // under-line의 width값을 under-line이 가지는 영역
만큼으로 바꾸겠다는 속성
    document.getElementById('under-line').style.top = '58px'
    document.getElementById('under-line').style.left = event.target.offsetLeft + 'px' // offsetLeft : 해당요소의 위치값 만큼 이동하는 속
    성

```

- **reload(새로고침)**

```

// location을 사용하는 방법
location.reload(); // Post 데이터를 포함해 페이지를 새로고침
location.replace(location.href); // Post 데이터는 포함하지 않으며 페이지를 새로 고침
// 이 때 현재 이력을 수정하여 페이지를 불러오기 때문에 history에 새로은 이력은 추가되지 않습니다.
location.href = location.href; // 페이지를 이동합니다. 이동 할 페이지를 현재 페이지로 지정함으로써 새로 고침처럼 화면이 호출됩니다.
// 페이지를 이동하기 때문에 history에 이력이 추가되며 엄밀히 말하면 페이지 새로고침은 아닙니다.

location.reload(true); // default: false
// 브라우저가 가지고 있는 기존의 리소스는 신경쓰지 않고 새로운 리소스를 받아 화면을 갱신합니다.

location === window.location // true

window.location === document.location // true

// location, window.location, document.location은 동일하기 때문에 브라우저간의 호환성 문제가 있어 location의 사용에 문제가 있는 경우 아래와 같은 방법으로 페이지를 새로 고침 할 수 있습니다.

function reload() {
    (location || window.location || document.location).reload();
}

// history를 사용하는 방법
history.go(0);

※ 특정 주기로 페이지 새로 고침
1. setInterval()을 이용해 페이지 새로 고침
setTimeout(function () {
    location.reload();
}, 60 * 1000);

2. meta 태그를 이용해 페이지 새로 고침
// 사용 구문 설명
<meta http-equiv='refresh' content='초; url=페이지'>

// 현재 페이지를 기준으로 10초마다 현재 페이지로 이동한다.
<meta http-equiv='refresh' content='10;'>

// 현재 페이지를 기준으로 10초뒤에 https://www.daum.net/ 페이지로 이동한다.
<meta http-equiv='refresh' content='10; url=https://www.daum.net/'>

※ 주의할 점 - 새로 고침이 안되는 경우
location을 이용해 페이지를 새로 고침하거나 이동하는 경우 동일 출처 정책에 따라 기능이 동작하지 않을 수도 있으니 위에서 제시된 코드를 사용해 페이지 새로 고침이 되지 않는 경우 해당 정책을 위반했는지 확인해보세요.

```

- **animate() = css animation 과 동일한 효과를 js에 적용**

```

1. 형식 #1
element.animate(keyframes, duration)

1-1. 예시 #1
window.addEventListener("load", function() {
    var el = document.querySelector("#box");
    el.onclick = function() {
        this.animate([
            { left: "0%", transform: "rotate(0deg)" },
            { left: "50%", transform: "rotate(360deg)" }
        ], 2000);
    });
});

2. 형식 #2
element.animate(keyframes, optional)

2-2. 예시 #2
window.addEventListener("load", function() {
    var el = document.querySelector("#box");
    el.onclick = function() {
        this.animate([
            { left: "0%", transform: "rotate(0deg)" },
            { left: "50%", transform: "rotate(360deg)" }
        ], {
            duration: 2000,
            fill: "forwards"
        });
    });
});

```

```

        });
    });
});

3-1. 매개변수 (Use of parameters 1)
- Array keyframes 필수

- 애니메이션 키프레임을 가지고 있는 배열이며 각 키프레임은 객체로 구현한다. 그리고 객체의 프로퍼티로 애니메이션이 적용될 속성을 정의한다.

- 키 프레임은 정의한 객체(속성 명과 값으로 구성)를 원소로 가지고 있는 배열
이 방식은 getKeyframes() 메소드가 반환한 표준 형식이다.

document.querySelector('.demo').animate([
    // from keyframe
    {
        opacity: 0,
        backgroundColor: #FFF
    },
    // to keyframe
    {
        opacity: 1,
        backgroundColor: #CCC
    }
], 1000);

- 오프셋 값을 설정하여 각 키프레임에 대한 오프셋을 지정할 수 있다. 오프셋은 타임라인상에서 구간을 설정하는 것이며 0~1사이의 값을 지정한다. 첫번째와 마지막 속성에서는 정의할 수 없다.
document.querySelector('.demo').animate([
    { transform: translateX(0px) },
    { transform: translateX(200px), offset: 0.7 },
    { transform: translateY(400px), offset: 0.8 },
    { transform: translateX(600px) }
], 2000);

- 키프레임 사이에 easing을 적용할 수 있다. 설정된 키프레임에서 다음 키프레임까지만 적용된다.
document.querySelector('.demo').animate([
    { transform: translateX(0px) },
    { transform: translateX(200px), easing: ease-out },
    { transform: translateY(400px), easing: ease-in },
    { transform: translateX(600px) }
], 2000);

- Object keyframes 필수
- 속성과 값이 배열로 구성된 key-value 쌍을 포함하는 객체
document.querySelector('.demo').animate({
    backgroundColor: ['#FFF', '#FFFF00'], // [from, to]
    color: ['#000', '#0000FF'] // [from, to]
}, 2000);

- 이 형식의 경우에는 다음 아래의 샘플 코드처럼 각 배열의 원소 수가 같을 필요는 없다. 지정된 값은 독립적으로 배치된다.
document.querySelector('.demo').animate({
    backgroundColor: ['#FFF', '#FFFF00', '#00FF00'], // offset: 0, 0.5, 1
    color: ['#000', '#0000FF'] // offset: 0, 1
}, 2000);

- 다음의 경우는 애니메이션 속성에 대한 단계의 값과 오프셋, easing을 원소로 갖는 배열로 설정했다.
document.querySelector('.demo').animate({
    backgroundColor: ['#FFF', '#FFFF00', '#00FF00],
    offset: [0, 0.3, 1],
    easing: [ease-in, ease-out]
}, 2000);

3-2. 매개변수 (Use of parameters 2)
- Integer duration 필수
- 애니메이션의 지속 시간을 밀리초(milliseconds)로 설정한다.

- Dictionary optional 필수
- 하나 이상의 옵션 속성을 가지고 있는 객체를 지정한다. 다음 아래와 같은 옵션을 사용할 수 있다.

id : 생략 가능하며 고유 식별자를 DOMString 으로 지정
delay : 애니메이션을 지연시키는 시간을 밀리초(milliseconds)로 설정한다. 생략 가능하며 기본 값은 0이다.

direction : 애니메이션의 실행 방향을 normal(정방향), reverse(역방향), alternate(정방향후 역방향), alternate-reverse(역방향후 정방향)로 설정할 수 있다.
기본 값은 normal 이다.

duration : 애니메이션이 완료되는 시간을 밀리초(milliseconds)로 설정한다. 선택사항이지만 기본값은 0이므로 설정하지 않는다면 애니메이션이 실행되지 않는다.

easing : 애니메이션의 속도 변화를 지정한다. 사전에 정의된 값은 linear, ease, ease-in, ease-out, ease-in-out로 지정할 수 있으며 cubic-bezier() 함수도 사용이 가능하다. 생략할 수 있으며 기본 값은 linear이다.

endDelay : 애니메이션이 끝난 후 지연되는 시간을 밀리초(milliseconds)로 설정할 수 있다. 다른 애니메이션의 종료 시간을 기준으로 애니메이션을 시퀀싱할 때 주로 사용 한다. 생략이 가능하며 기본 값은 0이다.

fill : 요소가 기본적으로 가지고 있는 style과 키프레임에 정의된 style과의 관계를 설정한다. backwards로 설정하면 바로 첫 키프레임의 style을 가지고 애니메이션이 시작되며 애니메이션이 완료되면 원래의 정의된 style로 돌아오지만 forwards로 설정하면 원래의 style을 유지하다가 애니메이션이 시작되며 애니메이션이 종료된 후 마지막 키프레임의 style을 유지한다. both는 시작 키프레임으로 style을 가지고 애니메이션을 시작하며 끝나면 마지막 키프레임의 style을 유지한다. 키프레임에 의해 유지된 값은

```

외부의 CSS 적용으로 재정의해도 그 값을 그대로 유지한다. 기본 값은 원래의 style을 가지고 애니메이션을 시작하고 끝나면 원래의 style로 돌아오는 none이다.

iterationStart : 반복의 어느 시점에서 애니메이션을 시작해야 하는지를 설정한다. 예를 들어 0.5는 첫번째 반복의 중간에서 시작을 나타내며 이 값을 설정하면 두번 반복하는 애니메이션이 세번째 반복 중 중간에서 종료된다. 생략이 가능하며 기본 값은 0.0이다.

iterations : 애니메이션이 반복되는 횟수를 지정한다. 기본 값은 1이며 요소가 존재하는 한 반복될 수 있도록 Infinity 키워드를 사용할 수도 있다.

```
document.getElementById('under-line').animate({'scale':['0', '1']},{durations:500,fill:'forwards',easing:'ease'})
```

[웹 컴파인 #4 : JQuery]

• 제이쿼리 기본

⇒ 자바스크립트 라이브러리 중 하나

⇒ 자바스크립트의 명령을 보다 간단하게 사용할 수 있도록 해준다

⇒ 제이쿼리 기본 문서를 반드시 html 문서에 연결해서 사용해야 한다

⇒ 내가 작성할 내용들 보다 앞에 작성되어야 한다

⇒ 제이쿼리는 버전이 1,2,3이 있다

⇒ 버전1과 2는 거의 동일한 문서인데, 버전 2가 HTML5 전용으로 만들어졌다 (현재는 의미없음)

⇒ 버전3은 CSS3의 애니 명령, 색상 명령 등이 추가되었고, 버전 2의 일부 명령이 삭제 또는 개정되었다

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>00_JQuery_시작</title>
</head>
<body>
    <h1>JQuery</h1>
    <textarea cols="100" rows="100">
        - 자바스크립트의 라이브러리 문서
        - 자바스크립트보다 간단하고 쉽게 사용 가능
        - 제이쿼리 명령은 사용 전 반드시 연결 필요
        - 기본 문법
        $(document).ready(function(){
            동작할 내용
        });
    </textarea>
    <script src="js/jquery-3.7.0.min.js"></script>
    <script>
        $(document).ready(function(){
            $('h1').css('color', 'red')
        })
    </script>
</body>
</html>
```

• 제이쿼리 셀렉터

1. 명령이 적용 될 대상

- 클릭해야 할 버튼, 팝업 창, 메뉴 목록, 슬라이드 등등등

2. 셀렉터의 형태 : \$('셀렉터')

3. css에서 사용했던 모든 형태의 셀렉터를 제이쿼리에서 그대로 사용 가능하다

- \$('.div') \$('.wrap') \$('.gnb') \$('.list li')

• 제이쿼리 명령

1. 명령의 가장 기본적인 형태 : \$('셀렉터').명령어();

2. 하나의 대상에 여러개의 명령이 순차적으로 실행될 때 명령을 사슬처럼 엮어서 사용 가능하다

- \$('셀렉터').명령어().명령어().명령어().명령어();
- 3. 명령어에 따라서 인자(인수)가 있을 수도 없을 수도 있다
- 4. 파라미터(매개변수)가 정의되어 진 명령의 경우, 인자를 전달할 때와 전달하지 않을 때의 결과가 달라질 수 있다
- 5. 여러개의 파라미터(매개변수)가 있는 명령도 존재할 수 있다

[JQuery 연결 #1]

- <https://code.jquery.com/jquery-3.7.0.min.js> 내용복사
- 파일생성 후 복사한 내용 붙여넣기
- jquery-릴리즈-버전.min.js 이름으로 저장 (관례)

※ <script>를 <body> 맨 뒤에 삽입하는 이유? : <https://lipcoder.tistory.com/500>

[JQuery 연결 #2]

※ 제이쿼리 다운로드 방법(http://www.tcpschool.com/jquery/jq_intro_apply)

- 1) 파일 다운로드 : [jQuery.com](http://jquery.com) : <http://jquery.com/download> =>

제이쿼리 파일을 다운받아 로드하는 방법

최신 버전의 제이쿼리 파일은 다음 공식 사이트에서 다운받을 수 있습니다.

[jQuery.com : http://jquery.com/download](http://jquery.com/download) =>

이렇게 다운받은 제이쿼리 파일을 서버에 저장하고, 다음 <script>태그를 웹 페이지의 <head> 태그 내에 삽입하면 됩니다.

문법

```
<head>
  <script src="/파일경로/제이쿼리파일명.js"></script>
</head>
```

예제

```
<head>
  <script src="/media/jquery-1.12.4.min.js"></script>
</head>
```

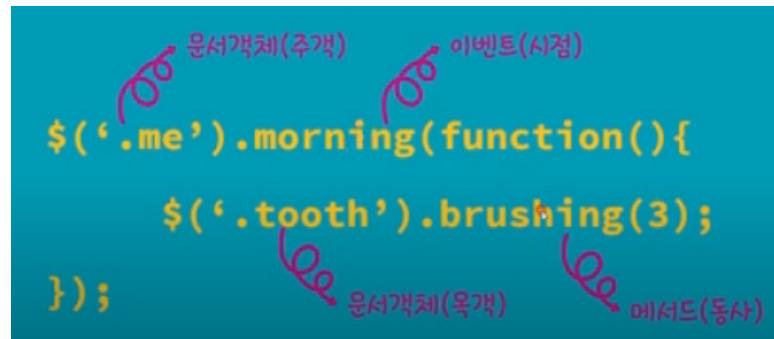
2) [CDN]

1. [jQuery.com](http://jquery.com) CDN : <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
2. 구글 CDN : <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
3. MS CDN : <script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.12.4.min.js"></script>
4. CDNJS CDN : <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
5. jsDelivr CDN : <script src="https://cdn.jsdelivr.net/jquery/1.12.4/jquery.min.js"></script>

※ 제이쿼리 css 적용 예시

```
$('h1').css({
    color : 'red'
})

document.querySelector('h1').style.color = 'blue'
```



※ 자바스크립트 기반 프레임워크 : vue, next, node

※ 자바스크립트 기반 라이브러리 : react

※ 제이쿼리를 사용하는 이유

- 아래코드는 왜? JQuery를 사용해야 되는지를 보여주는 예제이다

[HTML 코드]

```
<body>
    <ul class ="menu" id ="menu1">
        <li>menu1-1</li>
        <li>menu1-2</li>
        <li>menu1-3</li>
        <li>menu1-4</li>
        <li>menu1-5</li>
        <li>menu1-6</li>
        <li>menu1-7</li>
    </ul>
    <ul class ="menu" id ="menu2">
        <li>menu2-1</li>
        <li>menu2-2</li>
        <li>menu2-3</li>
        <li>menu2-4</li>
        <li>menu2-5</li>
    </ul>
```

[자바스크립트 코드]

```
window.onload = function(){
    let menu2 = document.getElementById("menu2");
    let liList = menu2.getElementsByTagName("li");
    for(let i = 0; i < liList.length; i++){
        let li = liList[i];
        li.style.color = "#f00";
    }
}
```

[제이쿼리 코드]

```
$(document).ready(function(){
    $("#menu2 li").css("color", "#f00");
})
```

Ex01. 지구본 움직이기 #1

- index.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ex01. 지구본 움직이기</title>
    <link rel="stylesheet" href="css/gigu.css">
</head>
<body>
    <div id="paner">
        <div id="bar"></div>
        
        <div id="btn">
            <button id="btnStart">지구본 움직이기</button>
        </div>
    </div>
</body>
```

```

        </div>
    </div>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="js/gugu.js"></script>
</body>
</html>

```

- common.css

```

@charset "UTF-8";

#paner{
    width: 600px;
    height: 300px;
    border: 1px solid #999;
    position: relative;
}

#bar{
    position: absolute;
    left: 50px;
    top: 170px;
    width: 500px;
    height: 15px;
    background-color: #f33;
}

#gigu{
    position: absolute;
    left: 50px;
    top: 80px;
}

#btn{
    text-align: center;
    width: 600px;
}

#btnStart{
    position: absolute;
    top: 300px;
    left: 250px;
    font-size: 14px;
    font-weight: bold;
    padding: 10px;
    margin-top: 10px;
}

```

- common.js

```

$(document).ready(function(){
    // 지구 이미지 웹 요소(노드) 저장
    let gigu = $("#gigu");

    // 버튼 클릭 이벤트
    // 5초에 거쳐 원쪽으로 430px 만큼 이동
    $('#btnStart').click(function(){
        gigu.animate({
            left : '480px'
        }, 5000);
    })
})

```

Ex02-1. 글자 보이기, 숨기기

- index.html

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ex02-1. 글자 숨기기, 보이기 예제</title>
</head>
<body>
    <div>안녕하세요. 초보 JQuery-3 버전입니다.</div>

```

```

<button id="showText">글자 보이기</button>
<button id="hideText">글자 숨기기</button>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="js/text.js"></script>
</body>
</html>

```

- **text.js**

```

$(document).ready(function(){

    // 최초 글자 숨긴 상태 설정
    $('div').hide();

    $('#showText').click(function(){
        $('div').show();
    })
    $('#hideText').click(function(){
        $('div').hide();
    })
})

```

Ex02-2. 지구본, 텍스트 흔들기

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ex02-2. 지구본, 텍스트 흔들기</title>
    <link rel="stylesheet" href="css/giguText.css">
</head>
<body>
    <div class="hello">
        <div class="text">안녕하세요!!</div>
    </div>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="js/giguText.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "UTF-8";

.hello{
    background: url(../../../Ex0/images/gigu.png) no-repeat center / contain;
    width: 400px;
    height: 400px;
    position: relative;
}

.hello .text{
    bottom: 0;
    color: brown;
    font-family: Verdana !important;
    font-size: 40pt !important;
    font-weight: bold;
    position: absolute;
    text-shadow: 0 0 5px rgba(0,0,0,0.5);
}

```

- **giguText.js**

```

function hello() {
    // 좌표값과 배경이미지의 크기를 난수로 대입하기 위한 값 할당
    let rnd1 = Math.floor(Math.random() * 20);           // 시작값 : 0 ~ 20
    let rnd2 = Math.floor(Math.random() * 40);           // 시작값 : 0 ~ 40
    let rnd3 = Math.floor(Math.random() * 3) + 100;      // 시작값 : 100 ~ 103
}

```

```

    // 글자흔들기
    $('.hello').css({
        bottom : rnd1,
        left : rnd2
    })
    // 배경 확대
    $('.hello .text').css({
        "background-size" : rnd3 + "%"
    })
}

// 0.01초마다 hello 함수 호출
setInterval(hello, 10)

```

Ex02-3. 지구본 움직이기 #2

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ex02-3. 지구본 움직이기</title>
    <link rel="stylesheet" href="css/giguMove.css">
</head>
<body>
    <div>
        <div id="panel">
            
        </div>
    </div>

    <div id="nav">
        <label>x 값 입력 : </label>
        <input id="txtX"><br>
        <label>y 값 입력 : </label>
        <input id="txtY"><br>
        <button id="btnStart">지구본 움직이기</button>
    </div>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="js/giguMove.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "UTF-8";

body{
    font-size: 10pt;
}

#panel{
    width: 600px;
    height: 500px;
    border: 1px solid #999;
    position: relative;
}

#gigu{
    position: absolute;
    width: 80px;
    left: 50px;
    top: 120px;
}

#nav{
    width: 600px;
    text-align: center;
}

```

- **giguMove.js**

```

// 움직임을 줄 요소(대상 - 지구본) 값 초기화
let gigu = null

```

```

$(document).ready(function(){
    initialize();
    event_gigu();
})

// 전역변수를 초기화 하는 함수
function initialize(){
    // 움직임을 줄 요소(대상 - 지구본) 선택
    gigu = $('#gigu')
}

// 이벤트와 관련된 내용을 함수로 등록
function event_gigu(){
    // 버튼 클릭 이벤트
    $('#btnStart').click(function(){
        // 지구본 움직이기
        // 지구본 위치 값 구하기
        let x = parseInt($('#txtX').val())
        let y = parseInt($('#txtY').val())
        // 위치값에 따라 지정된 요소가 움직임을 실행하는 함수
        moveGigu(x, y);
    });
}

// 지구본의 움직이는 기능 구현 함수
function moveGigu(x, y){
    // 대상이 움직이는 범위 영역 지정
    if((x >= 0 && x <= 500) && (y >= 0 && y <= 300)){
        gigu.css({
            left : x,
            top : y
        })
    }else {
        alert("입력된 값이 범위를 벗어났습니다.")
    }
}

```

Ex03. 메인페이지 클론코딩 예제(WHOIS MARKETING 사이트)

- index.html

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ex03. WHOIS MARKETING 메인페이지 클론코딩</title>
    <link rel="stylesheet" href="css/reset.css">
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="wrap">
        <header>
            <h1><a href="#"></a></h1>
            <div class="gnb">
                <ul>
                    <li><a href="#" class="mainnav">회사소개</a></li>
                    <li><a href="#" class="mainnav">바이럴 마케팅</a></li>
                    <li><a href="#" class="mainnav">신청&amp; 문의</a></li>
                    <li><a href="#" class="mainnav">고객센터</a></li>
                    <li><a href="#"></a></li>
                    <li><a href="#"></a></li>
                </ul>
            </div>
        </header>
        <!-- header_end -->
        <div class="main">
            <ul>
                <li>
                    <a href="#">
                        <p>시업 성공 파트너</p>
                        <h2><span class="keyword">WHOIS</span> Marketing Center</h2>
                        <span class="small">블로그, SNS, 키워드 검색, 바이럴마케팅 분야 <span class="keyword">고객만족도 1위</span></span>
                    </a>
                </li>
            </ul>
        </div>
        <!-- main_end -->
        <div class="contents">
            <div class="bwrap">
                <div class="banner">

```

```

<ul>
    <li>
        <a href="#">
            
            <div class="banner_txt">
                <h4>블로그 광고</h4>
                <p>
                    소비자의 관심 품목을<br>
                    포스팅하여 확산시키는 광고
                </p>
            </div>
        </a>
    </li>
    <li>
        <a href="#">
            
            <div class="banner_txt">
                <h4>SNS 광고</h4>
                <p>
                    소비자 친화적인 SNS통한 광고<br>
                    빠른 전파속도가 강점
                </p>
            </div>
        </a>
    </li>
    <li>
        <a href="#">
            
            <div class="banner_txt">
                <h4>키워드검색 광고</h4>
                <p>
                    정확한 소비자 타겟 설정으로<br>
                    성공률이 높인 광고
                </p>
            </div>
        </a>
    </li>
</ul>
</div>
<!-- banner_end -->
</div>
<!-- bwrap_end -->
<div class="cwrap">
    <h2>CONTACT US</h2>
    <p>소중한 정보를 남겨주시면 상담을 위한 신속한 연락 드리겠습니다</p>
    <div class="user_info_wrap">
        <h4>개인정보 수집 및 이용에 관한 안내</h4>
        <p>
            회사는 귀하께서 회사의 개인정보보호방침 또는 이용약관의 내용에 대해 『동의안함』 버튼을 출력할 수<br>
            있는 절차를 마련하여, 『동의한다』 버튼을 클릭하면 개인정보 수집에 대해 동의한 것으로 봅니다
        </p>
        <form class="info">
            <input type="radio" name="chk_info" value="동의함">동의함
            <input type="radio" name="chk_info" value="동의안함">동의안함
        </form>
        <div class="user_info_box">
            <div class="user_info">
                <form class="user_name">
                    <input type="text" placeholder="이름">
                </form>
                <form class="phone_number">
                    <input type="text" value="-">
                    <input type="text" value="-">
                    <input type="text" value="-">
                </form>
            </div>
            <a href="#">등록</a>
        </div>
    </div>
    <!-- cwrap_end -->
</div>
<!-- conent_end -->
<footer>
    <h2>
        WHOIS <span>MARKETING</span>
    </h2>
    <ul class="policy">
        <li><a href="#">이용약관</a></li>
        <li><a href="#">개인정보방침</a></li>
        <li><a href="#">이메일주소무단수집거부</a></li>
    </ul>
    <ul class="address">
        <li><a href="#">(우:08378) 서울특별시 구로구 디지털로 34길 43(구로동, 코오롱싸이언스밸리1차)</a></li>
        <li><a href="#">회사명 : (주)후이즈드림</a></li>
        <li><a href="#" class="last">대표이사 : 이청중</a></li>
    </ul>

```

```

<ul class="office_info">
    <li><a href="#">사업자등록번호 : 120-86-22942</a></li>
    <li><a href="#">Email : dream@whoisg.com</a></li>
    <li><a href="#">Tel : 010-1234-5678</a></li>
    <li><a href="#" class="last">Fax : 02-1234-5678</a></li>
</ul>
<p>Copyrigth © (주)후이즈 All rights reserved. Designed & Programmed by WHOIS </p>
</footer>
<!-- footer_end -->
</div>
<!-- wrap_end -->
</body>
</html>

```

• **style.css**

```

@charset "UTF-8";

.wrap{
    width: 100%;
    height: 100vh;
    min-width: 1600px;
    min-height: 1600px;
}

header{
    width: 100%;
    height: 80px;
    display: flex;
    justify-content: space-evenly;
    align-items: center;
    border-bottom: 1px solid #999;
}

.gnb ul{
    display: flex;
    gap: 40px;
    justify-content: space-between;
    align-items: center;
}

.gnb .mainnav{
    display: block;
    font-size: 16px;
    color: #666;
}

header > .join{
    background-color: red;
    display: flex;
    justify-content: space-between;
    align-items: center;
}
/* header_end */

.main{
    width: 100%;
    height: 700px;
}

.main > ul{
    width: 100%;
    height: 100%;
}

.main > ul > li{
    width: 100%;
    height: 100%;
    background: url(..../images/mainV.jpg) no-repeat center;
    background-size: cover;
    position: relative;
}

.main a{
    width: 100%;
    height: 30%;
    display: flex;
    flex-direction: column;
    position: absolute;
    top: 120px;
    left: 250px;
    line-height: 80px;
}

```

```

.main p{
    font-size: 40px;
}

.main h2{
    font-size: 60px;
    font-weight: normal;
}

.main .keyword{
    color:#F25797;
    font-weight: bold;
}

.main .small{
    font-weight: bold;
    font-size: 25px;
}
/* main_end */

.contents{
    width: 100%;
}

.contents > .bwrap{
    width: 100%;
    height: 360px;
    background-color: #eee;
    position: relative;
}

.bwrap > .banner{
    width: 100%;
    height: 100%;
    background-color: #eddede;
}

.banner > ul{
    width: 100%;
    height: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
    gap: 25px;
    position: absolute;
    bottom: 100px;
}

.banner > ul > li{
    background-color: #fff;
}

.banner a{
    width: 400px;
    height: 360px;
    display: flex;
    flex-direction: column;
    justify-content: flex-start;
    align-items: center;
}

.banner a > .banner_txt{
    padding: 20px 0;
    text-align: center;
}

.banner_txt > h4{
    margin-bottom: 10px;
    font-size: 18px;
}
/* bwrap_end */

.cwrap{
    width: 100%;
    height: 700px;
    background: url(..../images/contact_bg.jpg) no-repeat center / cover;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}

.cwrap > h2{
    color: #fff;
    font-size: 60px;
}

```

```

.cwrap > p{
    color: #fff;
    font-size: 25px;
    padding: 20px 0;
}

.user_info_wrap > h4{
    color: #fff;
    font-size: 25px;
    margin-bottom: 10px;
}

.user_info_wrap > p{
    padding: 20px;
    border: 2px solid #fff;
    color: #fff;
    font-size: 25px;
}

.user_info_wrap > .info{
    float: right;
    font-size: 25px;
    color: #fff;
    padding: 10px 0;
}

.user_info_wrap > .info > input{
    width: 18px;
    height: 20px;
    margin-right: 5px;
}

.user_info_wrap > .user_info_box{
    width: 100%;
    margin-top: 60px;
    display: flex;
    justify-content: space-between;
    color: #fff;
    text-align: center;
    font-size: 25px;
}

.user_info_box > .user_info{
    width: 82%;
    height: 40px;
}

.user_info > .user_name{
    width: 100%;
    height: 100%;
    margin-bottom: 10px;
}

.user_name > input{
    width: 100%;
    height: 100%;

}

.user_info > .phone_number{
    width: 100%;
    height: 40px;
}

.phone_number > input{
    width: 31.42%;
    height: 40px;
}

.user_info_box a{
    display: block;
    width: 17%;
    height: 90px;
    background-color: navy;
    text-align: center;
    line-height: 90px;
}

/* content_end */

footer{
    width: 100%;
    height: 300px;
    background-color: #222;
    display: flex;
    flex-direction: column;
    justify-content: center;
}

```

```

        align-items: center;
        color: #999;
    }

    footer > h2{
        font-size: 40px;
    }

    footer > h2 > span{
        font-weight: normal;
    }

    footer ul{
        display: flex;
        gap: 20px;
    }

    footer > .policy{
        padding: 20px 0;
        font-size: 25px;
    }

    footer > .address, .office_info{
        font-size: 22px;
    }

    .address a::after, .office_info a::after{
        content: "|";
        font-size: 18px;
        position: relative;
        top: -2px;
        left: 10px;
    }

    .address .last::after, .office_info .last::after{
        display: none;
    }

    footer p{
        font-size: 25px;
        padding: 15px 0;
    }

```

※ JQuery 과제 1 풀이

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>제이쿼리 과제 1</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Abrib+Fatface&family=Bruno+Ace&family=Jua&family=Russo+One&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="css/exam1_style.css">
</head>
<body>
    <div class="wrap">
        <h1>과제명 : s01.js ~ s04.js => JavaScript 로 바꿔보기</h1>
        <div class="box-wrap">
            <div class="box1" id="box-1">
                <p class="p1">JQuery</p>
                <p class="p2" id="p2">JavaScript</p>
            </div>
            <div class="box2" id="box-2">
                <p class="p3">JQuery</p>
                <p class="p4">배상원</p>
                <p class="p5"></p>
                <p class="p6" id="p_6">JavaScript</p>
                <p class="p7" id="p_7" style="color:red">JavaScript</p>
            </div>
            <div class="box3" id="box-3">CSS 적용 1</div>
            <div class="box4" id="box-4">CSS 적용 2</div>
            <div class="box5" id="box-5">
                <p id="p9">Class Name 추가 : 클릭하면 Class Name 알람 이벤트</p>
            </div>
            <div id="box-6">Class Name 삭제 : 클릭할 때마다 Class Name 추가/삭제 (toggle 기능)</div>
        </div>
    </div>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="js/exam1.js"></script>

```

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

```
----- 1. 태그 컬러 변경 -----
// JQuery
$('.p1').css({
    color : 'red'
})

// JavaScript
document.getElementById('p2').style.color = 'blue'

----- 2. 텍스트 변경 -----
// JQuery
let languageName = "제이쿼리"
$('.box2 .p3').text(languageName)

let p4 = $('.box2 .p4').text()
$('.box2 .p5').text(p4)

let userName = $('.box2 .p4').text()
$('.box2 .p4').html(`<strong>${userName}</strong>님 환영합니다`)

// JavaScript
document.getElementById('p_6').innerText
document.getElementById('p_7').innerHTML="Java"

----- 3. CSS 명령 -----
// JQuery
$('.box3').css({
    width: '100%',
    height: 50,
    color : 'navy',
    backgroundColor: 'yellow',
    border : '2px solid #000',
    marginTop : 20,
    textAlign: 'center',
    lineHeight : '50px',           // lineHeight의 경우 명확히 px단위를 써줘야함 부모의 높이값을 그대로 수치만 적을 경우 텍스트 세로 가운데 정렬 안됨
})

let box4 = $('.box4').css('display')
box4 = 'none'
if(box4 == 'none'){
    $('.box4').css({
        border: '5px solid yellow'
    })
} else {
    $('.box4').css({
        border : 0
    })
}

// JavaScript
let box4Text = document.getElementById('box-4')
box4Text.style.backgroundColor = "gold"
box4Text.style.color = "blue"
box4Text.style.fontWeight = "bolder"
box4Text.style.textAlign = "center"
box4Text.style.lineHeight = "70px"

let box4Display = document.getElementById('box-4')
if(box4Display.offsetWidth > 0 && box4Display.offsetHeight > 0){      // element.offsetWidth > 0 && element.offsetHeight > 0) 일경우 display가 적용되는 경우
    box4Display.style.color = 'green'          // 즉, offsetWidth && offsetHeight > 0 이라는 것은 요소가 자리를 차지하고 있는지를 판단하는 조건
} else {
    box4Display.style.color = 'blue'
}

----- 4. Class 추가 / 제거 -----
// JQuery
let box5 = $('.box5').css('display')
console.log(box5)
if(box5 == 'block'){
    $('.box5').addClass('on')
} else {
    $('.box5').removeClass('on')
}

$('#box-6').click(function(){
    if($(this).hasClass("clicked")){
        $(this).removeClass("clicked")
```

```

        }else {
            $(this).addClass("clicked")
        }
        let alertMsg = $('#box-6')[0].className
        alert(alertMsg)
    })

// JavaScript
let box5Add = document.getElementById('box-5')
box5Add.className += ' new'           // 클래스 이름 추가 시 스페이스도 추가해야 함
box5Add.classList.add('off')          // element.className += ' 이름'; 과 달리 스페이스를 추가해주지 않아도 됨
                                    // classList.add('name1', 'name2', 'name3' 등...) 한번에 여러개 클래스 추가 가능
box5Add.addEventListener('click' , getClassItemName)

function getClassItemName(){
    const CLASSLENGTH = document.getElementById('box-5').classList.length
    for(let i = 0; i < CLASSLENGTH; i++){
        alert(document.getElementById('box-5').classList.item(i));
    }
}

if(box5Add.classList.length == 1){
    box5Add.style.backgroundColor = 'black'
    box5Add.style.color = 'white'
} else if(box5Add.classList.length == 2){
    box5Add.style.backgroundColor = 'blue'
    box5Add.style.color = 'black'
} else if(box5Add.classList.length >= 3){
    box5Add.style.backgroundColor = 'pink'
    box5Add.style.color = 'red'
    box5Add.style.fontWeight = 'bolder'
}

```

- 순서명령

⇒ 자바스크립트의 첫 번째 순서를 뜻하는 값은 0이다
 ⇒ index 순서값이라고도 한다

- \$(셀렉터).index() : 셀렉터의 순서 값을 저장하는 방식
- \$(셀렉터).eq(인덱스 순서값)
 ⇒ \$('.gnb li') : 메뉴의 li태그가 모두 선택된다
 ⇒ \$('.gnb li').eq(3) : 메뉴의 li태그 중 인덱스 순서 값이 3(휴먼 센스로 4번째)인 li태그만 선택된다

※ 순서 명령 예시

- index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>05 순서명령</title>
    <link rel="stylesheet" href="css/style05.css">
</head>
<body>
    <div class="box">
        <div class="btn">
            <p class="on">탭 1</p>
            <p>탭 2</p>
            <p>탭 3</p>
        </div>
    </div>
    <div class="pannel">
        <div class="inner view">탭 1의 내용</div>
        <div class="inner">탭 2의 내용</div>
        <div class="inner">탭 3의 내용</div>
    </div>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="js/s05.js"></script>
</body>
</html>

```

- style.css

```
@charset "utf-8";  
  
.btn{  
    display: flex;  
}  
  
.btn p{  
    border: 1px solid #aaa;  
    border-right: 0;  
    padding: 10px 20px;  
}  
  
.btn p:last-child{  
    border-right: 1px solid #aaa;  
}  
  
.btn .on{  
    background-color: lightblue;  
}  
  
.inner{  
    display: none;  
    width: 300px;  
    padding: 30px;  
    background-color: lightcyan;  
}  
  
/* inner란 클래스명과 view라는 클래스명을 동시에 가진 요소 선택 클래스명 사이에 띄워쓰기 x */  
.inner.view{  
    display: block;  
}
```

- common.js

```
* 버튼을 클릭할 때 탭(tab)이 바뀌도록  
$('.btn p').click(function(){  
    // 변수 i에 현재 선택하는 요소에 대한 순서값 저장(0, 1, 2...)  
    let i = $(this).index()  
    console.log(i)  
  
    // 전체 대상을 비활성화 한뒤 클릭하는 대상에 'on'이라는 클래스 이름 추가  
    $('.btn p').removeClass('on')  
    // this 가 가르키는 대상은 'btn p' 클래스  
    $(this).addClass('on')  
    // 클릭하는 시점에 활성화 된 패널이 달 수 있으니 모든 패널을 비활성화 시킨다  
    $('.pannel .inner').removeClass('view')  
    // 패널 중 1번째 패널만 선택해서 활성화 시킨다  
    // 활성화 된 시점에 'view'라는 클래스가 추가됨과 동시에 .inner.view 조건에 충족한 요소만 활성화되어 '탭 i의 내용' 이라는 문구가 출력된다  
    $('.pannel .inner').eq(i).addClass('view')  
})
```

- 이벤트(event)

⇒ 명령이 실행 될 타이밍 : 문서를 열 때, 클릭할 때, 마우스 올릴 때, 스크롤이 발생할 때 등등...

- 이벤트 명령의 형태

```
$( '셀렉터' ).이벤트명령( 함수로 이벤트로 인해 일어날 일들을 묶어둠 )  
$( '셀렉터' ).이벤트명령(function(){  
    이벤트로 인해 일어날 명령  
})
```

- 주로 사용되는 이벤트 명령어

⇒ click, mouseover, mouseout, mouseenter, mouseleave, scroll, resize

- 이벤트 유형(상세)

1) 마우스 이벤트

속성	설명
click	사용자가 HTML 요소를 마우스로 눌렀을 때 이벤트가 발생합니다.
dblclick	사용자가 HTML 요소를 마우스로 두 번 눌렀을 때 이벤트가 발생합니다.
mousedown	사용자가 요소 위에서 마우스 버튼을 누르는 동안 이벤트가 발생합니다.
mousemove	사용자가 요소 위에서 마우스 포인터를 움직일 때 이벤트가 발생합니다.
mouseover	마우스 포인터가 요소 위로 옮겨질 때 이벤트가 발생합니다.
mouseout	마우스 포인터가 요소를 벗어날 때 이벤트가 발생합니다.
mouseup	사용자가 누르고 있던 마우스 버튼에서 손을 떼 때 이벤트가 발생합니다.

2) 키보드 이벤트

속성	설명
keypress	사용자가 키를 눌렀을 때 이벤트가 발생합니다.
keydown	사용자가 키를 누르는 동안 이벤트가 발생합니다.
keyup	사용자가 키에서 손을 떼 때 이벤트가 발생합니다.

3) 폼 이벤트

속성	설명
blur	폼 요소에 포커스를 잃었을 때 이벤트가 발생합니다.
change	목록이나 체크 상태 등이 변경되었을 때 이벤트가 발생합니다(<input/> , <select>, <textare>) 태그에서 사용합니다).</textare></select>
focus	폼 요소에 포커스가 놓였을 때 이벤트가 발생합니다(<label>, <select>, <textare>) 태그에서 사용합니다).</textare></select></label>
reset	폼이 다시 시작되었을 때 이벤트가 발생합니다.
submit	submit 버튼을 눌렀을 때 이벤트가 발생합니다.

4) 문서 로딩 이벤트

속성	설명
abort	웹 문서가 완전히 로딩되기 전에 불러오기를 멈췄을 때 이벤트가 발생합니다.
error	문서가 정확히 로딩되지 않았을 때 이벤트가 발생합니다.
load	문서 로딩이 끝나면 이벤트가 발생합니다.
resize	문서 화면 크기가 바뀌었을 때 이벤트가 발생합니다.
scroll	문서 화면이 스크롤되었을 때 이벤트가 발생합니다.
unload	문서를 벗어날 때 이벤트가 발생합니다.

※ 이벤트 예시

- index.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>06 이벤트</title>
    <link rel="stylesheet" href="css/style06.css">
</head>
<body>
    <h1>클릭 이벤트</h1>
    <div class="box1">
        <button class="bt11">CHANGE</button>
        <button class="bt12">RESET</button>
        <p></p>
    </div>
    <br><hr><br>
    <div class="box2">
        <ul>
            <li class="active">목록1</li>
            <li>목록2</li>
            <li>목록3</li>
            <li>목록4</li>
            <li>목록5</li>
        </ul>
    </div>
    <br><hr><br>
    <div class="box3">
        <a href="">CLICK</a>
        <p></p>
    </div>
    <br><hr>      <-- <hr> : 수평선 태그 -->
    <h1>호버 이벤트</h1>
    <div class="box4">
        <p class="count01">0</p>
        <div class="inner">
            <p class="count01">0</p>
        </div>
    </div>
    <br><hr><br>
    <div class="box5">
        <p class="count02">0</p>
    </div>
</body>
```

```

<div class="inner">
    <p class="count02">0</p>
</div>
</div>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="js/s06.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "utf-8";

.box1 p{
    width: 300px;
    height: 300px;
    background-color: lawngreen;
}

.box2 .active{
    color: coral;
}

.box3 p{
    width: 300px;
    height: 300px;
    background-color: cyan;
}

.box4, .box5{
    width: 300px;
    padding: 30px;
    background-color: salmon;
}

.inner{
    padding: 20px;
    background-color: lightyellow;
}

```

- **common.js**

```

클릭시 색상 변경 #1
$('.box1 .bt11').click(function(){
    $('.box1 p').css ({
        backgroundColor:"red"
    })
})

$('.box1 .bt12').click(function(){
    $('.box1 p').css ({
        backgroundColor:"lawngreen"
    })
})

클릭시 목록(1 ~ 5) 변경
$('.box2 li').click(function(){
    $('.box2 li').removeClass('active')
    $(this).addClass('active')
})
}

클릭 시 색상 변경 #2
$('.box3 a').click(function(){
    $('.box3 p').css({
        backgroundColor:"gold"
    })
    // a 태그의 경우 href 속성 때문에 링크된 주소 또는 새로고침이 적용되기 때문에 return false를 적용해줄 것
    // return false 는 해당 함수를 종료하기 때문에 현재 상태에서 속성을 적용할 수 있음 (새로고침 x)
    return false
})

마우스 커서 팝업 시 숫자 증가 (자식요소에 마우스 팝업시에도 동일하게 효과 적용)
선택된 요소의 범위 내에서 마우스를 이동해도 숫자 증가
let x = 0
$('.box4').mouseover(function(){
    $('.count01').text(++x)
})

마우스 커서 팝업 시 숫자 증가 (자식요소에서는 숫자 증가 x)

```

```

마우스를 요소의 범위에 진입해야만 숫자 증가(즉, 마우스 커서를 요소 밖으로 완전히 나갔다 들어와야만 숫자 증가)
let y = 0
$('.box5').mouseenter(function(){
    $('.count02').text(++y)
})

```

※ 제이쿼리 동적 기능

- **addClass()** : 요소 추가
- **removeClass()** : 요소 제거
- **toggleClass()** : 있으면 제거, 없으면 추가
- **hasClass()** : 특정 클래스를 가지고 있는지 여부 확인
- **text()** : 요소의 텍스트 추가 및 값 가져오기
- **css()** : CSS 속성 적용 및 값 가져오기

※ 클릭 시 Class Name 알림 이벤트 발생 예시

```

$('#box-6').click(function(){
    if($(this).hasClass("clicked")){
        $(this).removeClass("clicked")
    }else {
        $(this).addClass("clicked")
    }
    let alertMsg = $('#box-6')[0].className
    alert(alertMsg)
})

```

※ Class Name 존재 여부에 따라 추가 삭제 하는 기능 예시

```

$("#요소").click(function(){
    $(this).toggleClass("clicked");
});

```

※ 속성값(Attribute) 추가

1. 이미지 태그에 경로(src) 속성 추가
`\$('img').attr('src', 'javascript.png')`
2. 텍스트 속성 추가
`\$('h1').text('JavaScript 입니다')`
3. 태그 추가
`\$('h1').html('JavaScript 입니다.')`

※ mouseover 와 mouseenter 메서드의 차이 (참고 : <https://recoveryman.tistory.com/51>)



[이벤트 예제]

Ex01. COUPEL STORY 예제

- [index.html](#)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ex02. 커플 이벤트 데일리</title>
    <!-- <link rel="stylesheet" href="css/reset.css"> -->
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="navbar">
        
    </div>
    <!-- navbar_end -->

    <div class="main">
        <div class="container">
            <h2>만난지 <span id="day-count">00</span> 일째</h2>

            <div class="img-row">
                
                
                
                
                
                
                
            </div>
            <!-- img-row_end -->

            <!-- 시작일자 달력 삽입 -->
            <div class="date-row">
                <label for="start">시작일</label>
                <input type="date" name="start" id="start">
            </div>
            <!-- date-row_end -->
            <button id="calculate">결과확인</button>
        </div>
        <!-- container_end -->
    </div>
    <!-- main_end -->
    <script src="js/jquery-3.7.0.min.js"></script>
    <script src="js/Ex02.js"></script>
</body>
</html>
```

- [style.css](#)

```
body{
    margin: 0;
    min-width: 992px;
    font-family: "helvetica";
}
/* reset */

.navbar{
    height: 66px;
    background-color: #fff;
    text-align: center;
    line-height: 66px;
}

#logo{
    /* logo img 정렬 */
    vertical-align: middle;
}

.main{
    /* 그레이디언트 효과 적용 */
    background-image: linear-gradient(135deg, #B0A8FF 0%, #786FFF 100%);
    padding: 80px 0;
    min-height: calc(100% - 66px);
}

.container{
```

```

background-color: #fff;
box-shadow: 2px 2px 4px 2px rgba(0,0,0,0.1);
border-radius: 10px;
width: 600px;
margin: 0 auto;
text-align: center;
padding: 32px 66px;
}

.container > h2{
    font-size: 24px;
    color: #4a4a4a;
    margin: 0;
    margin-bottom: 28px;
}

.container > h2 > span{
    font-weight: 700;
    color: #789fff;
}

.container .img-row{
    margin-bottom: 40px;
}

.container .img-row img{
    vertical-align: middle;
    margin: 0 6px;
}

.date-row{
    margin-bottom: 40px;
}

```

- **Ex01.js**

```

/*
기본 Logic
1. 현재날짜와 시작날짜의 값을 저장하는 변수 생성
2. 현재날짜에서 시작날짜를 뺀 결과값 반환 하는 변수 생성
3. 반환된 결과값을 일(day) 단위로 변환
*/
$( '#calculate' ).on( 'click', calculateDate );

function calculateDate(){
    // 현재날짜 계산
    let now = new Date();
    console.log(now)

    // 시작날짜 계산
    let startDate = new Date( $( '#start' ).val() );
    console.log(startDate)

    // 결과값 반환 : (현재날짜 - 시작날짜) / sec / min / hour / day
    // 1000ms -> sec , 60sec -> min , 60 -> hour, 24hour -> day
    let betweenDate = (now.getTime() - startDate.getTime()) / 1000 / 60 / 60 / 24;
    console.log(betweenDate);

    // 반환된 결과값 반영
    // 현재날짜 1일의 경우 결과값 카운트에서 빼지기 때문에 +1 적용
    $( '#day-count' ).text( Math.floor(betweenDate) + 1 )

}

```

- **스크롤 이벤트**

- ⇒ 스크롤 이벤트는 스크롤의 현재 위치값을 비교해서 동작을 지정하는 경우가 많다
- ⇒ 스크롤의 이동거리 = 컨텐츠의 이동거리
- ⇒ 스크롤이 오른쪽 방향으로 10px 이동하면 컨텐츠도 왼쪽방향으로 10px 이동되는 것이 연속적으로 발생해서 움직이는 것처럼 보여진다
- ⇒ 현재 위치를 저장하는 방법 : scrollTop(), scrollLeft()

- ⇒ 실제 사이트 작업시에는 스크롤 이벤트가 발생될 시점의 숫자값을 정확하게 수동으로 입력할 수 없다
- ⇒ 컨텐츠의 유무에 따라 세로의 거리가 변동될 가능성이 높기 때문이다 PC, 테블릿, 노트북, 모바일에 따라서 환경이 변하기 때문에 정확한 숫자값을 알 수가 없다
- 그래서 이벤트가 발생될 스크롤의 위치값을 얻기 위해서 특정 태그의 절대 위치값을 활용해야 한다

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>스크롤</title>
    <style>
        body{
            font-family: "맑은 고딕", "고딕", "굴림";
            height: 2000px;
        }

        .top{
            display: inline-block;
            background-color: skyblue;
            height: 200px;
            width: 100vw;
            text-align: center;
            font-size: 42px;
            line-height: 200px;
            vertical-align: middle;
        }

        button{
            position: fixed;
            left: 1000px;
            top: 300px;
        }
    </style>
</head>
<body>
    <div class="top">
        인생은 아름다워~
    </div>

    <button class="go-to-top">Top</button>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3OJU5yExlq6GSYGSKh7tPXikynS7ogEvDej/m4=" crossorigin>
        // 위 -> 아래로 스크롤을 내렸을 때 위치값 출력
        $(window).on('scroll', function(){
            console.log($(window).scrollTop())
        })

        // 스크롤을 내릴 때마다 'opacity(투명도)' 조절
        $('.top').css('opacity', 1 - $(window).scrollTop() / $('.top').height())
    ) // 1 - 0 / 200 = 1 - 0 = 1
    // 100일 경우 : 1 - 100 / 200 = 1 - 0.5 = 0.5 반투명

    // 'Top' 버튼 클릭 시 0.5초에 걸쳐 페이지 상단으로 스크롤 이동
    $('.go-to-top').on('click', function(){
        $('html, body').animate({
            scrollTop:0
        }, 500)
        // $(window).scrollTop(0)
    })
</script>
</body>
</html>

```

* 위 → 아래로 스크롤을 내렸을 때의 위치 값



※ 절대 위치값이란? 문서가 시작되는 위치부터 특정 태그까지의 거리

- 절대 위치값을 저장하는 명령 : offset()
- offset() : 가로와 세로의 위치값을 한번에 저장한다
- offset().top : 세로의 위치값을 저장한다
- offset().left : 가로의 위치값을 저장한다

※ 스크롤 이벤트 예시

- **index.html**

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>07 스크롤 이벤트</title>
    <link rel="stylesheet" href="css/style07.css">
</head>
<body>
    <h1>스크롤 이벤트</h1>
    <div class="box1">
        <p>
            
        </p>
    </div>
    <br>
    <div class="box2">
        <div class="inner">
            <div>1</div>
            <div>2</div>
            <div>3</div>
            <div>4</div>
            <div>5</div>
        </div>
    </div>
    <br>
    <div class="box3">
        <div class="inner">
            <div class="b31">1</div>
            <div class="b32">2</div>
            <div class="b33">
                <span class="b33_icon">
                    
                </span>
            </div>
            <div class="b34">4</div>
            <div class="b35">5</div>
        </div>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="js/s07.js"></script>
</body>
</html>
```

- **style.css**

```

@charset "utf-8";

.box1{
    width: 300px;
    padding: 20px;
    background-color: antiquewhite;
    overflow-x: scroll;
}

.box2{
    width: 400px;
    background-color: lightblue;
    overflow-x: scroll;
}

.box2 .inner{
    display: flex;
    width: 1000px;
    padding: 50px;
}

.box2 .inner div{
    width: 200px;
    height: 200px;
}

/* nth-child 요소 중 홀수번째 해당하는 요소 */
.box2 .inner div:nth-child(odd){
    background-color: red;
}

/* nth-child 요소 중 짝수번째 해당하는 요소 */
.box2 .inner div:nth-child(even){
    background-color: greenyellow;
}

.box3{
    width: 300px;
    height: 300px;
    background-color: cadetblue;
    overflow-y: scroll;
}

.box3 .inner{
    padding: 50px;
}

.box3 .inner div{
    width: 200px;
    height: 200px;
}

.box3 .inner div:nth-child(odd){
    background-color: lightpink;
}
.box3 .inner div:nth-child(even){
    background-color: lightyellow;
}

.box3 .b33{
    overflow: hidden;
}

.box3 .b33 .b33_icon{
    display: block;
    transform: translateX(-200px);
    transition: .2s;
}

.box3 .b33.active .b33_icon{
    transform: translateX(0);
}

```

- **common.js**

```

좌 -> 우측으로의 스크롤의 움직임에 따라 배경 색상 변경
$('.box1').scroll(function(){
    $(this).css({
        backgroundColor: "red"
    })
})

```

```

좌 -> 우측으로의 스크롤의 움직임이 250px를 넘어가는 순간 배경 색상 변경
$('.box2').scroll(function(){
    let s = $(this).scrollLeft()
    if(s >= 250){
        $(this).css({
            backgroundColor : "yellow"
        })
    } else {
        $(this).css({
            backgroundColor : "lightsteelblue"
        })
    }
})

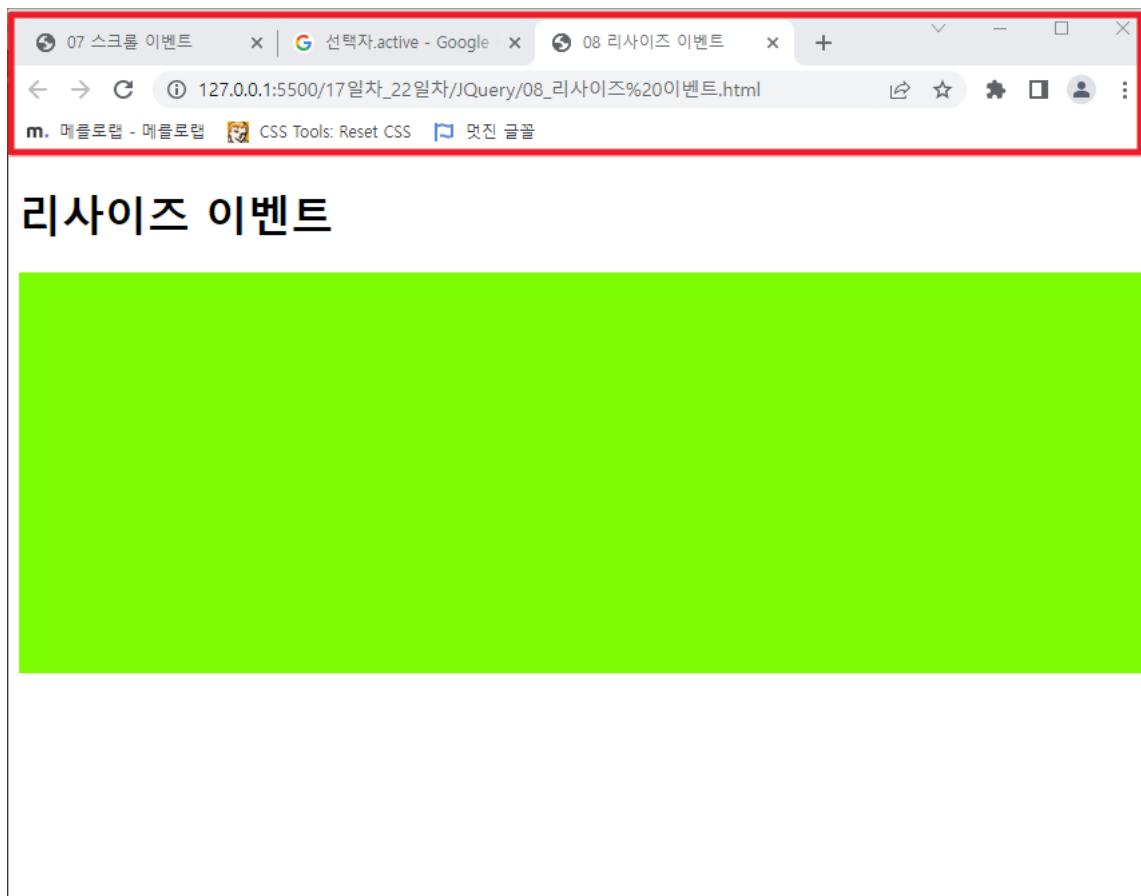
위 -> 아래로의 스크롤의 움직임이 350px를 넘어가는 순간 'active'클래스 추가, 이미지(b33_icon) 사진을 x축으로 -200px <-> 0px 이동
$('.box3').scroll(function(){
    let s = $(this).scrollTop()
    if(s >= 350){
        $('.b33').addClass('active')
    } else {
        $('.b33').removeClass('active')
    }
})

브라우저 스크롤 이벤트
Javascript에서 브라우저는 window라고 표현되어진다
$(window).scroll(function(){
    let i = $(window).scrollTop()
    // offset : target이 가지고 있는 고유 정보
    let b1 = $('.box1').offset().top
    let b2 = $('.box2').offset().top
    if(i >= b2){
        $('body').css({
            backgroundColor : "green"
        })
    } else if(i >= b1){
        $('body').css({
            backgroundColor : "blue"
        })
    } else {
        $('body').css({
            backgroundColor : "#ffff"
        })
    }
})

```

- 리사이즈 이벤트

※ `$(window).height` 의 경우 주소표시줄을 포함한 최상단 높이를 염두해 두어야 함



※ 리사이즈 이벤트 예시

- **index.html**

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>08 리사이즈 이벤트</title>
    <link rel="stylesheet" href="css/style08.css">
</head>
<body>
    <h1>리사이즈 이벤트</h1>
    <div class="box1"></div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="js/s08.js"></script>
</body>
</html>
```

- **style.css**

```
@charset "utf-8";

.box1{
    height: 300px;
    background-color: lawngreen;
}
```

- **common.js**

```
$(window).resize(function(){
    b1BgColor()
})
```

```

// 통상적인 모니터 해상도 : 1920 x 1080
function b1BgColor(){
    let i = $(window).width()
    if(i >= 1000){
        $('.box1').css({
            backgroundColor : "red"
        })
    } else {
        $('.box1').css({
            backgroundColor : "blue"
        })
    }
}

```

Ex02. LaunchPad

- index.html

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ex07. LaunchPad</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="pad">
        
        <button id="play-btn">
            
        </button>
        <button id="stop-btn">
            
        </button>

        <div class="cell-container">
            <div class="cell green" id="cell1">
                <div class="key">1</div>
                <div class="instrument">Bass<br>KickDrum</div>
            </div>
            <!-- cell1_end -->

            <div class="cell blue" id="cell2">
                <div class="key">2</div>
                <div class="instrument">Closed<br>Hi-hat</div>
            </div>
            <!-- cell2_end -->

            <div class="cell blue" id="cell3">
                <div class="key">3</div>
                <div class="instrument">Open<br>Hi-hat</div>
            </div>
            <!-- cell3_end -->

            <div class="cell red" id="cell4">
                <div class="key">4</div>
                <div class="instrument">Clap</div>
            </div>
            <!-- cell4_end -->

            <div class="cell red" id="cell5">
                <div class="key">5</div>
                <div class="instrument">Snap</div>
            </div>
            <!-- cell5_end -->

            <div class="cell blue" id="cell6">
                <div class="key">6</div>
                <div class="instrument">Crash<br>cymbal</div>
            </div>
            <!-- cell6_end -->

            <div class="cell green" id="cell7">
                <div class="key">7</div>
                <div class="instrument">Mid<br>Tom</div>
            </div>
            <!-- cell7_end -->

            <div class="cell green" id="cell8">
                <div class="key">8</div>
                <div class="instrument">Hi<br>Tom</div>
            </div>
        </div>
    </div>

```

```

        </div>
        <!-- cell8_end -->

        <div class="cell blue" id="cell9">
            <div class="key">9</div>
            <div class="instrument">Tambourine</div>
        </div>
        <!-- cell9_end -->
    </div>
    <!-- cell-container_end -->

</div>
<!-- pad_end -->

<!--
    - 구글크롬에서는 기본적으로 미디어플레이어에 대한 AutoPlay(자동재생) 속성을 지원하지 않는다
    src : 재생할 음원 파일의 경로를 설정(mp3, wav, ogg, ....)
    autoplay : 자동 재생 여부 설정(브라우저에 html 파일이 로드되면 음악파일이 재생됨)
    loop : 반복 재생 여부 설정
    controls : 컨트롤 패널 노출 여부
    muted : 음소거 설정
-->

생성, 재생, 정지
Audio 객체 이벤트
- canplaythrough : 음원 파일이 모드 로드되어 재생 가능할 때
- play : 재생이 시작될 때
- playing : 재생 중일 때
- pause : 일시 정지되었을 때
- ended : 재생이 완료되었을 때
- volumechange : 볼륨이 변경될 때

Audio 객체의 메소드와 프로퍼티
1) 메소드
play() : 재생
pause() : 일시정지
addTextTrack() : 새로운 트랙을 추가
canPlayType() : 브라우저가 해당 오디오 타입에 대한 재생 가능 여부 체크
load() : 오디오 객체를 리로드

2) 프로퍼티
src : 파일 경로
volume : 볼륨
loop : 반복여부(true, false)
autoplay : 자동재생여부(true, false)
muted : 음소거 여부(true, false)
paused : 일시정지 여부(true, false)
ended : 재생완료 여부(true, false)
duration : 음원의 전체 길이(초 단위)
currentTime : 음원의 현재 재생 위치(초 단위)

-->
<!-- <audio src="Queencard.mp3" controls></audio>

<input type="text" id="input"> -->
</body>
</html>

```

• style.css

```

*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body{
    margin: 0;
    font-family: "Helvetica";
    background-color: #090910;
}

.pad{
    width: 700px;
    height: 570px;
    margin: 100px auto;
    position: relative;
    border: 2px solid #ccc;
    border-radius: 4px;
}

.pad #play-btn{
    position: absolute;
    top: 70px;
}

```

```

        right: 50px;
        padding: 0;
        border: none;
        background-color: transparent;
        cursor: pointer;
        outline: none;
    }

.pad #stop-btn{
    position: absolute;
    top: 140px;
    right: 50px;
    padding: 0;
    border: none;
    background-color: transparent;
    cursor: pointer;
    outline: none;
}

.cell-container{
    width: 450px;
    height: 450px;
    left: 80px;
    top: 80px;
    position: absolute;
    background-color: #333;
    padding: 40px;
    border-radius: 4px;
}

.cell{
    width: 100px;
    height: 100px;
    color: #4A4A4A;
    font-size: 16px;
    float: left;
    margin: 10px;
    text-align: center;
    border-radius: 4px;
}

.cell .key{
    border: 2px solid #4a4a4a;
    border-radius: 50%;
    width: 40px;
    height: 40px;
    margin: 8px auto;
    line-height: 40px;
    font-weight: 500;
}

.cell .instrument{
    width: 80px;
    white-space: nowrap;
    margin: 0 auto;
    font-weight: 500;
}

.cell.green{
    background-color: #87e682;
}

.cell.blue{
    background-color: #2dbaf2;
}

.cell.red{
    background-color: #fb718c;
}

.cell.green.playing{
    background-color: rgba(155, 230, 130, 0.2);
    color: #87e682;
    border: 2px solid #87e682;
}

.cell.green.playing .key{
    border: 2px solid #87e682;
}

.cell.blue.playing{
    background-color: rgba(45, 186, 242, 0.2);
    color: #2dbaf2;
    border: 2px solid #2dbaf2;
}

.cell.blue.playing .key{

```

```

        border: 2px solid#2dbaf2;
    }

.cell.red.playing{
    background-color: rgba(251, 113, 140, 0.2);
    color: #fb718c;
    border: 2px solid #fb718c;
}

.cell.red.playing .key{
    border: 2px solid #fb718c;
}

```

- **Ex02.js**

```

// 각 키값에 해당하는 오디오 파일을 담기위한 배열 생성
let audioFiles = []
// audioFiles.push(new Audio('audio/01_I-AM.mp3'))
audioFiles.push(new Audio('audio/02_Spicy.mp3'))
audioFiles.push(new Audio('audio/03_love.mp3'))
audioFiles.push(new Audio('audio/04_Hype Boy.mp3'))
audioFiles.push(new Audio('audio/05_After LIKE.mp3'))
audioFiles.push(new Audio('audio/06_Teddy Bear.mp3'))
audioFiles.push(new Audio('audio/07_TOMBOY.mp3'))
audioFiles.push(new Audio('audio/08_Kitsch.mp3'))
audioFiles.push(new Audio('audio/09_Nxde.mp3')) */

audioFiles.push(new Audio('audio/Bass-Drum-1.wav'))
audioFiles.push(new Audio('audio/Bass-Drum-2.wav'))
audioFiles.push(new Audio('audio/Bass-Drum-3.wav'))
audioFiles.push(new Audio('audio/Bass-Drum-4.wav'))
audioFiles.push(new Audio('audio/Bass-Drum-5.wav'))
audioFiles.push(new Audio('audio/Bass-Drum-6.wav'))
audioFiles.push(new Audio('audio/Bass-Drum-7.wav'))
audioFiles.push(new Audio('audio/Bass-Drum-8.wav'))
audioFiles.push(new Audio('audio/Bass-Drum-9.wav'))

// 오디오 객체 생성
let loop = new Audio("Alan_Walker - The_Drum.mp3")

// play 버튼 클릭 시 음원 재생
$('#play-btn').on('click' , function(){
loop.play();
})

// stop 버튼 클릭 시 음원 멈춤 -> 다시 play 버튼을 눌렀을 때 처음부터 재생되도록
$('#stop-btn').on('click' , function(){
loop.pause();
loop.currentTime = 0;
})

// 키보드로 눌러진 키값에 해당하는 box에 playing 클래스 추가 후 css 스타일 변경
$(document).on('keydown' , function(e){
    // console.log(e.key)

    // 눌러진 키가 1~9 외 키일 경우 내부의 실행 구문 동작 x
    if((e.key) >= 1 && (e.key) <= 9){
        $('#cell'+ e.key).addClass('playing');
        // Number : 가져오는 값이 문자인지 숫자인지 명확하지 않을 경우 변환해주기위해 사용
        // 키보드로 눌러진 키값에 해당하는 box에 audioFiles 배열에 담겨 있는 요소(소리파일) 0번째부터 가져오기
        let cur = audioFiles[number(e.key) - 1];
        // 새로운 값이 입력됨과 동시에 값을 초기화(연속성으로 키보드를 누를 때 소리가 즉시 끊기지 않는 문제 해결)
        cur.currentTime = 0;
        cur.play()
        // console.log(cur);
    }
})

// 눌러진 키보드에 손을 떼면 키값에 해당하는 box에 playing 클래스 삭제 후 css 스타일 변경
$(document).on('keyup' , function(){
$('.cell').removeClass('playing')
})

```

- 구글크롬에서는 기본적으로 미디어플레이어에 대한 AutoPlay(자동재생) 속성을 지원하지 않는다

```

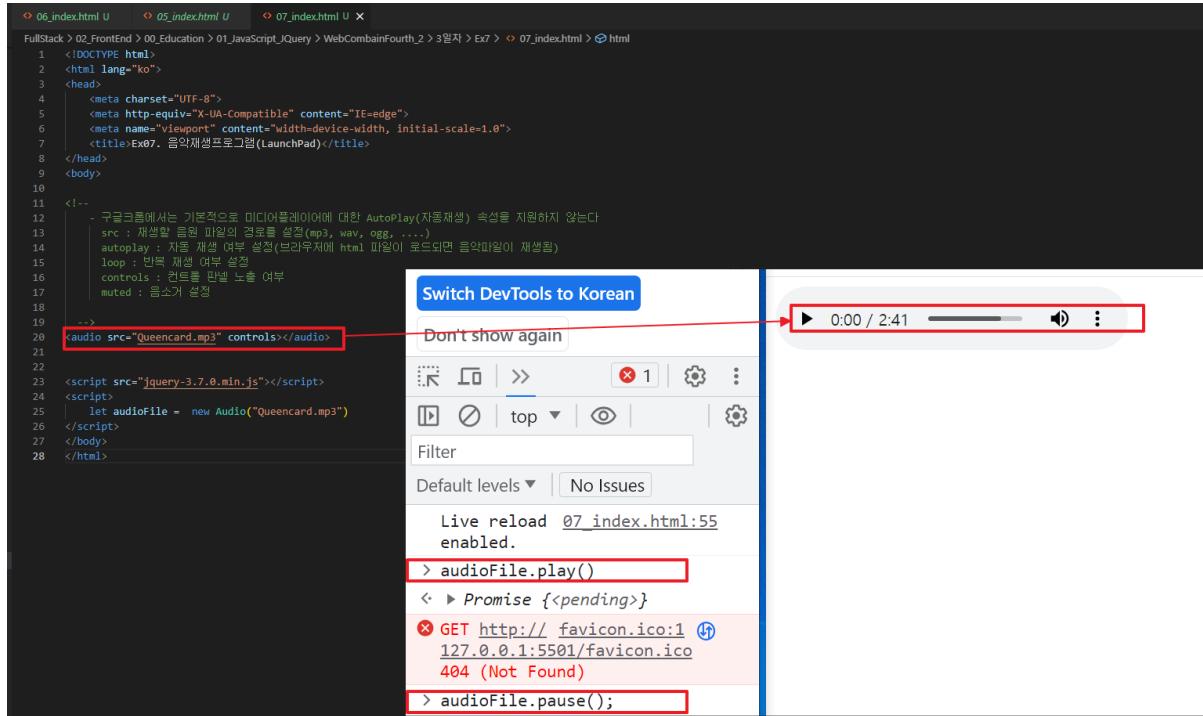
<audio src="Queencard.mp3" controls></audio>

* src : 재생할 음원 파일의 경로를 설정(mp3, wav, ogg, ....)
* 속성
- autoplay : 자동 재생 여부 설정(브라우저에 html 파일이 로드되면 음악파일이 재생됨)
- loop : 반복 재생 여부 설정

```

- controls : 컨트롤 패널 노출 여부
- muted : 음소거 설정

※ html과 js에서의 미디어 플레이어 사용



• Audio 객체 이벤트

- ⇒ canplaythrough : 음원 파일이 모드 로드되어 재생 가능할 때
- ⇒ play : 재생이 시작될 때
- ⇒ playing : 재생 중일 때
- ⇒ pause : 일시 정지되었을 때
- ⇒ ended : 재생이 완료되었을 때
- ⇒ volumechange : 볼륨이 변경될 때

• Audio 객체의 메소드와 프로퍼티

1) 메소드

- play() : 재생
- pause() : 일시정지
- addTextTrack() : 새로운 트랙을 추가
- canPlayType() : 브라우저가 해당 오디오 타입에 대한 재생 가능 여부 체크
- load() : 오디오 객체를 리로드

2) 프로퍼티

- src : 파일 경로
- volume : 볼륨
- loop : 반복여부(true, false)

- autoplay : 자동재생여부(true, false)
- muted : 음소거 여부(true, false)
- paused : 일시정지 여부(true, false)
- ended : 재생완료 여부(true, false)
- duration : 음원의 전체 길이(초 단위)
- currentTime : 음원의 현재 재생 위치(초 단위)

• 키보드 이벤트

⇒ keydown : 키보드 버튼을 누르는 순간
 ⇒ keypress : 키보드 버튼을 누르는 순간
 ⇒ keyup : 키보드 버튼을 놓렸다 땐 순간

• 키보드 이벤트 프로퍼티

⇒ keyboardEvent.key : 이벤트가 발생한 버튼의 값
 ⇒ keyboardEvent.type : 이벤트가 발생한 버튼의 키보드에서 물리적인 위치
 ⇒ code : 키보드에서 어느 위치에 있는 키를 눌렀는지 알려주는 것

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>키보드 이벤트 예시코드</title>
</head>
<body>
  <input type="text" id="input">
</body>

<script>
  let myInput = document.querySelector('#input');
  myInput.addEventListener('keydown', onkeyDown);

  function onkeyDown(e){
    // 누르는 키보드에 키에 대한 keyCode값 전달
    console.log(`키 : ${e.keyCode}`);
  }
</script>
</html>
```

• 키보드 이벤트 주의사항

⇒ keypress는 알파벳이나 숫자, 스페이스바 띄워쓰기 하는 것과 같이 출력값이 변하는 키에서는 이벤트가 발생하고, ESC, Shift 처럼 기능적인 역할을 하는 키들은 이벤트가 발생하지 않는다

⇒ 출력이 가능한 키라고 할지라도 한글 입력의 경우는 이벤트가 발생하지 않는다

※ 웹 표준에서 권장하는 방법은 keydown 이벤트 사용 권장

※ 키보드 이벤트 예시 (누르는 키보드 버튼의 고유 키 값 확인)

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>키보드 이벤트</title>
</head>
<body>
  <input type="text" id="input">

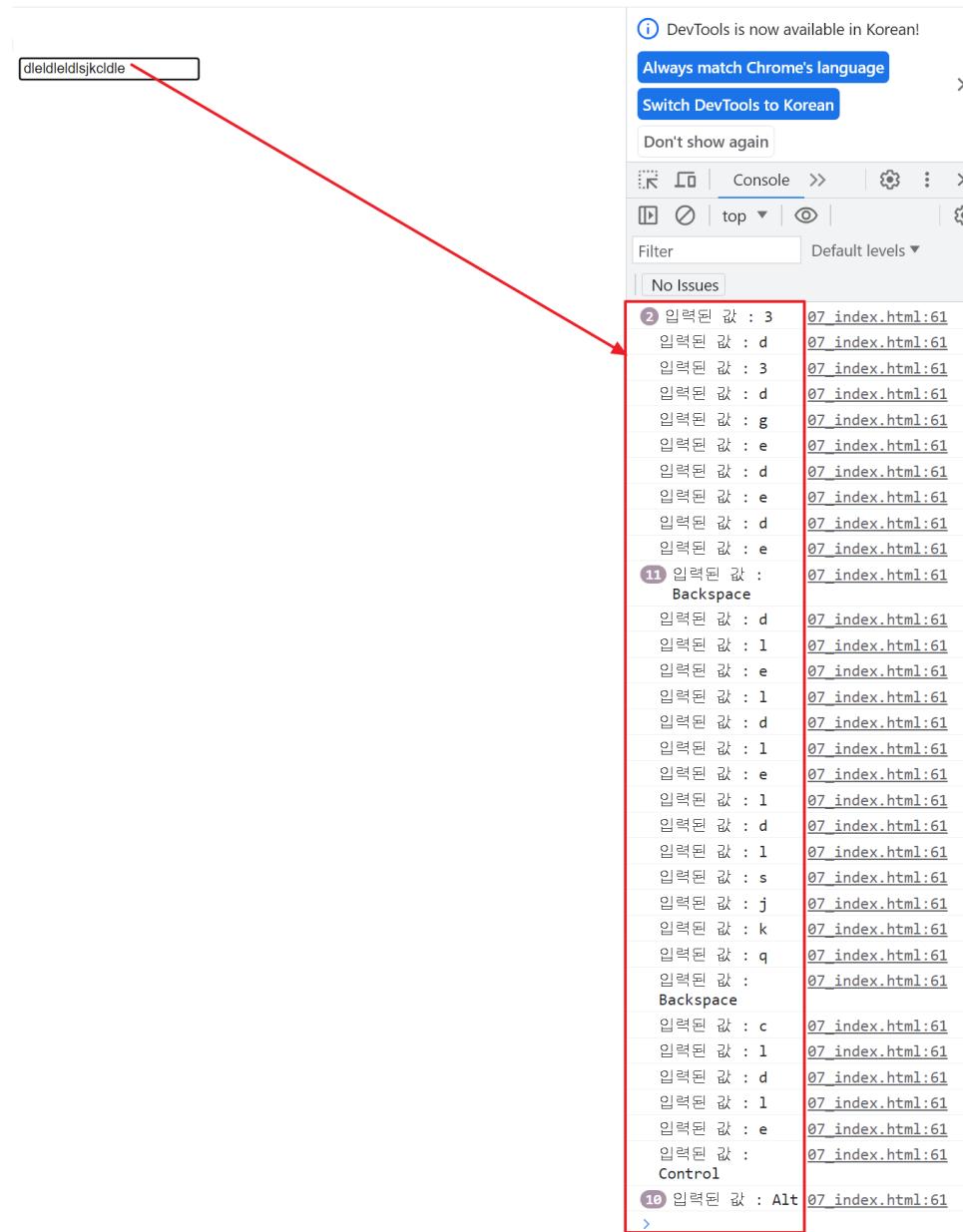
<script src="jquery-3.7.0.min.js"></script>
```

```

<script>
  const myInput = document.querySelector('#input');
  myInput.addEventListener('keydown', onKeyDown);

  function onKeyDown(e){
    console.log(`입력된 값 : ${e.key}`);
  }
</script>
</body>
</html>

```



Ex03. 청기백기 게임

- index.html

```

<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ex01. 청기백기게임</title>
  <link rel="stylesheet" href="css/style.css">

```

```

</head>
<body>
  <div class="container">
    <div class="flags">
      
      
      
      
      
      
    </div>
    <!-- flags_end -->

    <div class="buttons">
      <button id="btn1">청기 내려</button>
      <button id="btn2">청기 내리지 말고 백기 내려</button>
      <button id="btn3">점선 청기 내려</button>
    </div>
    <!-- buttons_end -->

  </div>
  <!-- container_end -->

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
  <script src="js/Ex.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "UTF-8";

body{
  width: 100%;
  height: 100vh;
  background: url(..../images/background.jpg) no-repeat center / cover;
}

.container{
  width: 100%;
  min-width: 1000px;
  height: 600px;
  margin: 60px auto;
  padding: 20px;
  text-align: center;
}

.flags{
  width: 100%;
  height: 40%;
}

.flags img{
  margin-right: 30px;
}

.buttons{
  width: 100%;
  height: 40px;
}

button{
  width: 200px;
  height: 100%;
  border-radius: 10px;
  border: none;
  font-size: 14px;
  margin-right: 30px;
  font-weight: bold;
}

.flags .flag.down{
  transform: rotate(30deg);
}

```

- **Ex03.js**

```

/*
기본 Logic
1) 버튼 클릭시 지정된 요소에 'down' Class 부여
2) 'flag' 와 'down' 이름을 동시에 가진 요소 30도 회전
3) 클릭 후 일정 시간이 지나면 'down' Class 삭제
4) 버튼이 클릭된 후 1초 뒤 원위치 하도록
*/

$( '#btn1' ).on( 'click', blueDown );
$( '#btn2' ).on( 'click', whiteDown );
$( '#btn3' ).on( 'click', blueDotDown );

function blueDown(){
    $('.blue').addClass('down')

    // 1초 뒤 원위치
    setTimeout(reset, 1000)
}

function whiteDown(){
    $('.white').addClass('down')
    setTimeout(reset, 1000)
}

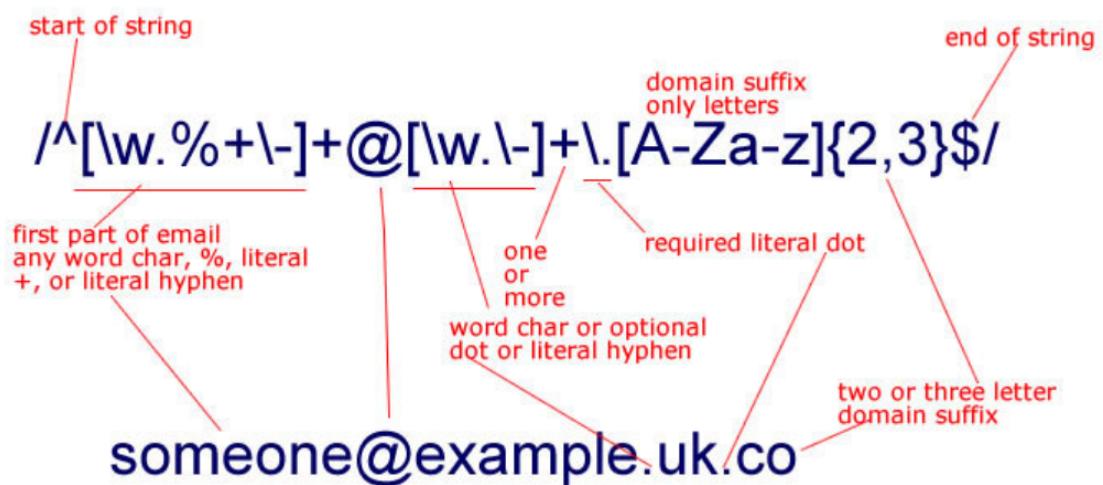
function blueDotDown(){
    $('.blue.dot').addClass('down')
    setTimeout(reset, 1000)
}

// 움직임이 발생한 깃발을 원위치 하는 함수
function reset(){
    $('.flag').removeClass('down')
}

```

Ex04. 카카오톡 채팅 창 클론 코딩

- 정규표현식



표현식	의미
<code>^x</code>	문자열의 시작을 표현하며 x 문자로 시작됨을 의미한다.
<code>x\$</code>	문자열의 종료를 표현하며 x 문자로 종료됨을 의미한다.
<code>.x</code>	임의의 한 문자의 자리수를 표현하며 문자열이 x로 끝난다는 것을 의미한다.
<code>x+</code>	반복을 표현하며 x 문자가 한번 이상 반복됨을 의미한다.
<code>x?</code>	존재여부를 표현하며 x 문자가 존재할 수도, 존재하지 않을 수도 있음을 의미한다.
<code>x*</code>	반복여부를 표현하며 x 문자가 0번 또는 그 이상 반복됨을 의미한다.
<code>x y</code>	or 를 표현하며 x 또는 y 문자가 존재함을 의미한다.
<code>(x)</code>	그룹을 표현하며 x를 그룹으로 처리함을 의미한다.
<code>(x)(y)</code>	그룹들의 집합을 표현하며 앞에서 부터 순서대로 번호를 부여하여 관리하고 x, y는 각 그룹의 데이터로 관리된다.
<code>(x)(?:y)</code>	그룹들의 집합에 대한 예외를 표현하며 그룹 집합으로 관리되지 않음을 의미한다.
<code>x{n}</code>	반복을 표현하며 x 문자가 n번 반복됨을 의미한다.
<code>x{n,}</code>	반복을 표현하며 x 문자가 n번 이상 반복됨을 의미한다.
<code>x{n,m}</code>	반복을 표현하며 x 문자가 최소 n번 이상 최대 m 번 이하로 반복됨을 의미한다.

표현식	의미
<code>[xy]</code>	문자 선택을 표현하며 x와 y 중에 하나를 의미한다.
<code>[^xy]</code>	not 을 표현하며 x 및 y를 제외한 문자를 의미한다.
<code>[x-z]</code>	range를 표현하며 x ~ z 사이의 문자를 의미한다.
<code>\^</code>	escape 를 표현하며 ^를 문자로 사용함을 의미한다.
<code>\b</code>	word boundary를 표현하며 문자와 공백사이의 문자를 의미한다.
<code>\B</code>	non word boundary를 표현하며 문자와 공백사이가 아닌 문자를 의미한다.
<code>\d</code>	digit 를 표현하며 숫자를 의미한다.
<code>\D</code>	non digit 를 표현하며 숫자가 아닌 것을 의미한다.
<code>\s</code>	space 를 표현하며 공백 문자를 의미한다.
<code>\S</code>	non space를 표현하며 공백 문자가 아닌 것을 의미한다.
<code>\t</code>	tab 을 표현하며 탭 문자를 의미한다.
<code>\v</code>	vertical tab을 표현하며 수직 탭(?) 문자를 의미한다.
<code>\w</code>	word 를 표현하며 알파벳 + 숫자 + _ 중의 한 문자임을 의미한다.
<code>\W</code>	non word를 표현하며 알파벳 + 숫자 + _ 가 아닌 문자를 의미한다.

Flag	의미
<code>g</code>	Global 을 표현하며 대상 문자열내에 모든 패턴들을 검색하는 것을 의미한다.
<code>i</code>	Ignore case 를 표현하며 대상 문자열에 대해서 대/소문자를 식별하지 않는 것을 의미 한다.
<code>m</code>	Multi line을 표현하며 대상 문자열이 다중 라인의 문자열인 경우에도 검색하는 것을 의미한다.

- 정규표현식 사용 예시

- 개별 숫자
/[0-9]/g : 전체에서 0~9사이 숫자 중 아무 숫자 '하나' 찾음
- 개별 문자
/[tō]/g : 전체에서 t 혹은 o를 모두 찾음
- 단어
/filter/g : 전체에서 'filter'라는 단어에 매칭되는 것을 찾음
- 단어 제외
/\b(?!bto\b)\w+wb : 전체에서 'to'라는 단어를 빼고 다른 단어 매칭
- 이메일
/^[\w+\.-\w+@\w+\.-\w+\.]\w+\$/i
: 시작을 0~9 사이 숫자 or a-z, A-Z 알파벳 아무거나로 시작하고 / 중간에 - _ . 같은 문자가 있을 수도 있고 없을 수도 있으며 / 그 후에 0~9 숫자 or a-z, A-Z 알파벳
- 전화번호
/^\d{3}-\d{3,4}-\d{4}\$/: 시작을 숫자 01로 시작하며 그 후에 0, 1, 6, 7, 8, 9 중에 하나가 나올수도 있으며 / 하이픈 - 하나 존재할수도 있으며 / 숫자 3~4개 이어지
- URL
^((https?):\/\/(([^:\.\\\s]+)(([^:\\/]*))?)?((\\([^\s\\/]+)*))?)?\\/(\\([^\#\\s]?)*)(\\(?(([^\\#\\s]*))?)?#((\\w*))?)\$
: ^((https?):\/\/(([^:\.\\\s]+)(([^:\\/]*))?)?((\\([^\s\\/]+)*))?)?\\/(\\([^\#\\s]?)*)(\\(?(([^\\#\\s]*))?)?#((\\w*))?)\$

- index.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>cocotok</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="chat-container">
        <div class="chat-header">
            <div class="btn_wrap">
                <button id="close" class="header-btn"></button>
                <button id="minimize" class="header-btn"></button>
                <button id="maximize" class="header-btn"></button>
            </div>
            <div class="user-info">
                
                <span id="username">여자친구</span>
            </div>
        </div>
        <div class="chatbox">
            <div class="friend-bubble bubble">
                자기야~~♡ 뭐해~?!
            </div>
            <div class="friend-bubble bubble">
                2023년 늘 변함없이 사랑해줘서 고마워
            </div>
            <div class="friend-bubble bubble">
                보고싶어~♡
            </div>
        </div>
        <!-- chatbox_end -->

        <div class="text-box">
            <textarea placeholder="대화내용이 들어갑니다!"></textarea>
            <button id="send">전송</button>
            <div class="clearfix"></div>
        </div>
        <!-- text-box_end -->
    </div>
    <!-- chat-container_end -->

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="js/cocotok.js"></script>
</body>
</html>
```

※ `div width` 사이즈 자동 조절

1. `display` 속성 변경 : `inline-block`;
-> 단, 이 경우는 필요에 따라 줄바꿈 속성(ex `
`)을 사용해주어야 한다.
`.bubble{`

```

/* display: inline-block; */
height: 30px;
line-height: 30px;
background-color: #fff;
border-radius: 10px;
margin-bottom: 10px;
padding: 0 10px;
clear: both;
float: left;
}

2. clear:both; float:left 속성 적용
-> 줄바꿈 속성을 추가로 사용하지 않고 element의 크기만큼 width영역을 할당한다.

```

- **style.css**

```

@charset "UTF-8";

.chat-container{
    width: 100%;
    min-width: 800px;
    height: 100vh;
    background: url(..../images/background.jpg) no-repeat center / cover;
}

.chat-header{
    width: 100%;
    height: 80px;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
}

.chat-header .header-btn{
    width: 15px;
    height: 15px;
    border-radius: 50%;
    border: none;
}

#close{
    background-color: red;
}

#minimize{
    background-color: orange;
}

#maximize{
    background-color: lightgreen;
}

.user-info{
    width: 100%;
    height: 60px;
    display: flex;
    align-items: center;
    gap: 10px;
}

#profile-plc{
    border-radius: 50%;
}
/* chat-header_end */

.chatbox{
    width: 100%;
    height: 800px;
    background-color:#D7E3F0;
    padding: 20px;
}

.bubble{
    /* display: inline-block; */
    height: 30px;
    line-height: 30px;
    background-color: #fff;
    border-radius: 10px;
    margin-bottom: 10px;
    padding: 0 10px;
    clear: both;
    float: left;
}
/* chatbox_end */

```

```

.text-box{
    width: 100%;
    height: 100px;
    padding: 10px;
    display: flex;
    align-items: center;
    gap: 10px;
}

textarea{
    width: 80%;
    height: 80px;
    border-radius: 10px;
    padding: 10px;
    font-size: 16px;
}

#send{
    width: 15%;
    height: 100px;
    background-color: #4590DB;
    color: #fff;
    font-size: 18px;
    font-weight: bold;
    border-radius: 10px;
    border: none;
}

.my-bubble {
    background-color: gold;
    clear: both;
    float: right;
    margin-right: 20px;
}

```

- **cocotok.js**

```

// '전송' 버튼 클릭시 발생하는 이벤트
$('#send').on('click', sendmybubble);

// 'Enter' 키를 누를 경우 '전송' 버튼을 클릭한 것과 동일한 기능을 실행하도록 하는 이벤트
$('textarea').on('keyup', sendmessage);

function sendmessage(e){
    let key = e['key'];
    // console.log(key)
    let val = $('textarea').val();
    if(key == 'Enter' && !e.shiftKey){
        sendmybubble();
    }
}

function sendmybubble(){
    // trim() : 앞뒤의 공백요소 제거
    let message = $('textarea').val().trim();
    // 정규표현식, (/(\n|\r\n)/g, '<br>') : 모든 문자열을 줄바꿈 해줌
    let result = message.replace(/(\n|\r\n)/g, '<br>');

    // 공백이 아닌 경우 전송
    if(message != " "){
        $('.chatbox').append('<div class="my-bubble bubble">' + result + '</div>');
        $('textarea').val(' ');
    }else {
        $('textarea').val(' ');
    }
}

```

- 제이쿼리에서 요소 더 선택하기

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>요소 더 선택하기</title>
    <style>
        box{
            display: flex;

```

```
        flex-wrap: wrap;
        margin: 20px auto;
        max-width: 23rem;
        border: 2px solid #f6774f;
    }

    button{
        border: none;
        color: white;
        margin: 1rem;
        width: 150px;
        height: 40px;
    }

    .color-1{
        background-color: #426fc5;
    }

    .color-2{
        background-color: #00897b;
    }

    .color-3{
        background-color: #f6774f;
    }

    .color-4{
        background-color: #e94043
    }

```

</style>

</head>

<body>

```
    <div class="box" id="box-1">
        <button class="color-1"></button>
        <button class="color-2"></button>
        <button class="color-3"></button>
        <button class="color-4"></button>
    </div>
    <div class="box" id="box-2">
        <button class="color-1"></button>
        <button class="color-2"></button>
        <button class="color-3"></button>
        <button class="color-4"></button>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

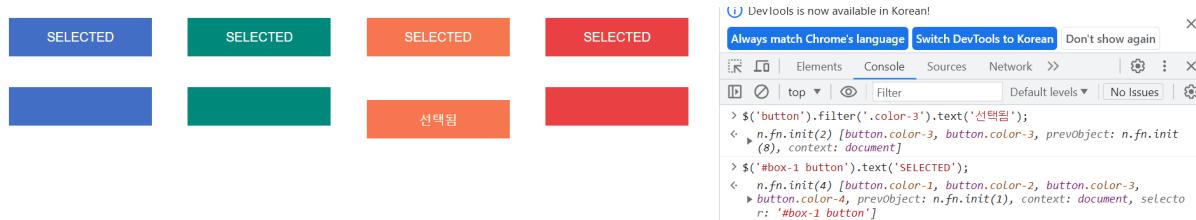
```

1) filter : 조건에 충족하는 요소만 걸러서 선택

```
$('.button').filter('.color-3').text('선택됨');

$('#box-1 button').text('SELECTED');

$('#box-2 button').filter('.color-3').text('SELECTED');
```



2) not : 선택한 요소만 제외

```
$( 'button' ).not( '.color-3' ).text( 'SELECTED' );
```

```
$('button').not('.color-3').text('SELECTED');
< n.fn.init(6) [button.color-1, button.color-2, button.color-4, prevObject: n.fn.init(8), context: document]
  > 0: button.color-1
  > 1: button.color-2
  > 2: button.color-4
  > 3: button.color-1
  > 4: button.color-2
  > 5: button.color-4
  > context: document
  length: 6
  prevObject: n.fn.init(8) [button.color-1, button.color-2, button.color-3, button.color-4, prevObject: n.fn.init(8), context: document]
  [[Prototype]]: Object(0)
```

3) eq : 선택한 요소들 중에서 N번째 요소

```
($('button').eq(1).text('SELECTED');
```



```
< DevTools is now available in Korean!
Always match Chrome's language Switch DevTools to Korean Don't show again
Elements Console Sources Network >
Default levels ▾ No Issues
> $('button').eq(1).text('SELECTED');
< n.fn.init [button.color-2, prevObject: n.fn.init(8), context: document]
```

4) parent : 선택한 요소의 부모 요소를 찾아준다

```
$('#box-1 .color-1').parent().css('background-color', 'black');
```



```
< DevTools is now available in Korean!
Always match Chrome's language Switch DevTools to Korean Don't show again
Elements Console Sources Network >
Default levels ▾ No Issues
> $('#box-1 .color-1').parent().css('background-color', 'black');
< n.fn.init [div#box-1.box, prevObject: n.fn.init(1), context: document]
> $('#box-2 .color-1').parent().css('background-color', 'gold');
< n.fn.init [div#box-2.box, prevObject: n.fn.init(1), context: document]
```

5) children : 선택한 요소의 자식 요소를 찾아준다

```
$('#box-1').children().css('background-color', 'black');
```



```
< DevTools is now available in Korean!
Always match Chrome's language Switch DevTools to Korean Don't show again
Elements Console Sources Network >
Default levels ▾ No Issues
> $('#box-1').children().css('background-color', 'black');
< n.fn.init(4) [button.color-1, button.color-2, button.color-3, button.color-4, prevObject: n.fn.init(1), context: document]
```

6) find : 부모, 자식 상관 없이 내가 원하는 요소를 선택할 때 사용

```
$('#box-1').find('.color-2').css('background-color', 'black');
```



7) siblings : 형제 요소를 찾아준다

```
$('#box-1 .color-2').siblings().text('SELECTED');

$('#box-2 .color-3').siblings('.color-4').text('SELECTED');
```



- **캡슐화 : 코드의 간소화**

⇒ 변경 전

```
<!DOCTYPE html>
<html>
<head>
<title>Travel Website</title>
<meta charset="utf-8">
<link href="css/style.css" rel="stylesheet">
</head>
<body>

<div id="menu">
<a id="home">Home</a>
<a id="seoul">Seoul</a>
<a id="tokyo">Tokyo</a>
<a id="paris">Paris</a>
</div>

<script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3OJU5yExlq6GSYGHk7tPXikynS7ogEvDej/m4=" crossorigin>
$(#home).on('click', clickHome);
$(#seoul).on('click', clickSeoul);
$(#tokyo).on('click', clickTokyo);
$(#paris).on('click', clickParis);

$('#menu a').on('click', selectMenu);

$(document).on('keydown', processKeyEvent);

function clickHome() {
  $('#photo').attr('src', 'images/home.png');
  $('#home').css('font-weight', 'bold');
  $('#seoul').css('font-weight', 'normal');
  $('#tokyo').css('font-weight', 'normal');
  $('#paris').css('font-weight', 'normal');
}

function clickSeoul() {
  $('#photo').attr('src', 'images/seoul.png');
  $('#seoul').css('font-weight', 'bold');
  $('#home').css('font-weight', 'normal');
  $('#tokyo').css('font-weight', 'normal');
  $('#paris').css('font-weight', 'normal');
}

function clickTokyo() {
  $('#photo').attr('src', 'images/tokyo.png');
  $('#tokyo').css('font-weight', 'bold');
  $('#home').css('font-weight', 'normal');
  $('#seoul').css('font-weight', 'normal');
  $('#paris').css('font-weight', 'normal');
}
```

```

        function clickParis() {
            $('#photo').attr('src', 'images/paris.png');
            $('#paris').css('font-weight', 'bold');
            $('#home').css('font-weight', 'normal');
            $('#tokyo').css('font-weight', 'normal');
            $('#seoul').css('font-weight', 'normal');
        }

        function mouseEnterPhoto() {
            $('#photo').css('box-shadow', '5px 10px');
        }

        function mouseLeavePhoto() {
            $('#photo').css('box-shadow', 'none');
        }

        function processKeyEvent(event) {
            if (event['key'] === '1') {
                clickHome();
            }
            else if (event['key'] === '2') {
                clickSeoul();
            }
            else if (event['key'] === '3') {
                clickTokyo();
            }
            else if (event['key'] === '4') {
                clickParis();
            }
        }
    </script>
</body>
</html>

```

⇒ 변경 후

```

<!DOCTYPE html>
<html>
<head>
    <title>Travel Website</title>
    <meta charset="utf-8">
    <link href="css/style.css" rel="stylesheet">
</head>
<body>
    
    <div id="menu">
        <a id="home">Home</a>
        <a id="seoul">Seoul</a>
        <a id="tokyo">Tokyo</a>
        <a id="paris">Paris</a>
    </div>
    
    <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3OJU5yExlq6GSYGHk7tPXikynS7ogEvDej/m4=" crossorigin>
        $('#menu a').on('click', selectMenu);
        $(document).on('keydown', selectMenu);

        // mouse hover 효과 이벤트
        $('#photo').on('mouseenter', mouseEnterPhoto);
        $('#photo').on('mouseleave', mouseLeavePhoto);

        // 키보드 1,2,3,4 key와 매칭되는 이미지 전환
        function selectMenu(e){
            let targetId = '';
            if(e.type === 'click'){
                targetId = e.currentTarget.id;
            }
            else if(e.type === 'keydown'){
                if(e.key === '1'){
                    targetId = 'home';
                }
                else if (e.key === '2'){
                    targetId = 'seoul';
                }
                else if (e.key === '3'){
                    targetId = 'tokyo';
                }
                else if (e.key === '4'){
                    targetId = 'paris'
                }
            }
        }
    
```

※fadeIn/Out 효과 적용

```

1) 최초 이미지 '숨김상태' 적용
$('#photo').hide

// 선택된 요소와 매칭된 이미지 변경
$('#photo').attr('src', 'images/' + targetId + '.png');

2) 이미지 보이기
$('#photo').fadeIn(1000)

// 메뉴의 a태그 모두 폰트 굵기 '보통' 상태로 변경
$('#menu a').css('font-weight', 'normal');

// 선택한 요소만 폰트 굵기 '굵게' 상태로 변경
$('#' + targetId).css('font-weight', 'bold')

}

// 마우스 hover 시 boxt 그림자 효과
function mouseEnterPhoto() {
    $('#photo').css('box-shadow', '5px 10px');
}

function mouseLeavePhoto() {
    $('#photo').css('box-shadow', 'none');
}

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

```

Ex. 쇼핑 사이트 팝업창 이벤트

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>FURNITURE SHOP</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="navbar">
        <a href="#" id="logo">
            
        </a>

        <ul id="menu">
            <li><a href="#">SHOP</a></li>
            <li><a href="#">CART</a></li>
            <li><a href="#">LOGIN</a></li>
        </ul>
    </div>
    <!-- navbar_end -->

    <div class="hero-header">
        <div class="info">
            <h1>코리아 페스타<br>슈퍼 세일</h1>
            <h2>최대 50% 할인의 혜택을 받아보세요!</h2>
            <button id="popup-trigger">쿠폰 받기</button>
        </div>
        <!-- info_end -->
    </div>
    <!-- hero-header_end -->

    <div id="popup" role="alert">
        <div id="popup-container">
            <h3>다운 완료!</h3>
            <p>내 쿠폰함에서 확인하세요!</p>

            <button id="close-btn">확인</button>
        </div>
        <!-- popup-container_end -->
    </div>
    <!-- popup_end -->

    <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3OJU5yExlq6GSYGSKh7tPXikynS70gEvDej/m4=" crossorigin="anonymous"></script>
    <script src="FS.js"></script>
</body>
</html>

```

- **style.css**

```
@charset "UTF-8";  
  
*{  
    margin: 0;  
    padding: 0;  
    box-sizing: border-box;  
    font-size: 14px;  
    color: #333;  
}  
  
li{  
    list-style: none;  
}  
  
a{  
    text-decoration: none;  
    color: initial;  
}  
/* reset */  
  
body{  
    width: 100%;  
    height: 100vh;  
    min-width: 1200px;  
    min-height: 700px;  
    background: url(images/bg.jpg) no-repeat center / cover;  
}  
  
.navbar{  
    width: 100%;  
    height: 100px;  
    background-color: #fff;  
    display: flex;  
    justify-content: space-between;  
    align-items: center;  
    padding: 0 20px;  
}  
  
.menu{  
    width: calc(100% - 600px);  
    display: flex;  
    justify-content: space-between;  
}  
  
.menu a{  
    font-size: 16px;  
    font-weight: bold;  
}  
  
.hero-header{  
    width: 100%;  
    height: 600px;  
}  
  
.info{  
    width: 100%;  
    height: 100%;  
    padding: 100px 0;  
    position: relative;  
}  
  
.info > h1{  
    font-size: 50px;  
    position: absolute;  
    top: 50%;  
    left: 50%;  
    transform: translate(-50%, -50%);  
    text-shadow: 2px 2px 2px rgba(255,255,255,0.7);  
}  
  
.info > h2{  
    font-size: 20px;  
    position: absolute;  
    top: 65%;  
    left: 50%;  
    transform: translate(-45%, -50%);  
    text-shadow: 2px 2px 2px rgba(255,255,255,0.7);  
    font-weight: bold;  
}  
  
.info > #popup-trigger{  
    position: absolute;
```

```

        top: 70%;
        left: 38%;
        width: 150px;
        height: 50px;
        background-color: #866248;
        color: #fff;
        font-weight: bold;
        border: none;
        border-radius: 10px;
        font-size: 18px;
    }

    #popup{
        display: none;
        position: absolute;
        width: 100%;
        height: 100vh;
        background-color: rgba(0,0,0,0.8);
        position: fixed;
        top: 0;
        left: 0;
        z-index: 1000;
    }

    #popup-container{
        width: 400px;
        height: 200px;
        background-color: #fff;
        border-radius: 10px;
        display: flex;
        flex-direction: column;
        justify-content: space-evenly;
        align-items: center;
        position: absolute;
        top: 150px;
        left: 0;
        right: 0;
        margin: 0 auto;
    }

    #popup-container > h3{
        font-size: 20px;
        font-weight: bold;
    }

    #popup-container > p{
        font-size: 20px;
        color: #666;
    }

    #popup-container > #close-btn{
        width: 150px;
        height: 50px;
        background-color: #866248;
        border: none;
        border-radius: 30px;
        color: #fff;
        font-weight: bold;
        cursor: pointer;
    }
}

```

- **js**

```

// '쿠폰받기' 버튼 클릭 시 팝업창 팝업
$('#popup-trigger').click(function(){
    $('#popup').fadeIn(800)
})

// '확인' 버튼 클릭 시 팝업창 닫기
$('#close-btn').click(function(){
    $('#popup').fadeOut(0)
})

// 키보드 'ESC'키 누르면 팝업창 닫기
$(document).keydown(function(e){
    if(e.which == '27'){
        $('#popup').fadeOut(0)
    }
})

```

Ex. 홀수 / 짝수 판별

- **each()** : 배열, Map, 그리고 객체를 매개변수로 받아, 마치 반복문처럼 그 요소들을 검사하고 반복할 수 있도록 하는 함수

※ each() 함수의 사용

```
// 1. 배열
$.each( array, callback );

each() 메서드는 첫번째 매개변수로 넘어온 배열의 0번 index 요소부터 한번씩 순회할 때마다 콜백함수를 실행합니다.
첫번째 인수로 배열이 입력되면, 콜백함수는 순서대로 index와 값을 매개변수로 갖게됩니다.
var arr = [1, 3, 5, 7]; // 배열 선언

// each() 메서드의 첫번째 매개변수로 위에서 선언한 배열을 전달
$.each(arr, function(index, value){
    console.log(index + " : " + value);
})

/* 실행 결과
0 : 1
1 : 3
2 : 5
3 : 7
*/

var arr = [
    { name: 'James', age: 25 },
    { name: 'Lucy', age: 24 }
];

$.each(arr, function (index, value) {
    var result = '';
    result += index + ' : ' + value.name + ', ' + value.age;
    console.log(result);
})

// 출력 결과
// 0 : James, 25
// 1 : Lucy, 24

// 2. 객체
$.each( object, callback );

each() 메서드의 첫번째 인수로 객체나 Map 등을 넘겨주는 경우에는 속성, 함수, 메서드를 포함한 모든 멤버를 반복하여 순회합니다. 이때 콜백함수의 매개변수는 순서대로 key(property), value, object가 됩니다.
var sampleObj = {
    name : "John",
    age : 28,
    printInfo : function(){
        console.log(name + ', ' + age);
    }
};

$.each(sampleObj, function(attrName, attrValue){
    console.log(attrName + " : " + attrValue);
})

// 출력 결과
// name : John
// age : 28
// printInfo : function(){
//     console.log(name + ', ' + age);
// }

// 3. Selector로 얻은 DOM 객체
$(selector).(callback);

jQuery 의 selector로 DOM 객체를 얻어와 each() 메서드에서 사용할 수 있습니다.
아래 예제에서는 listClass의 모든 <li>태그를 selector로 잡아서 each() 메서드를 실행하고 있습니다.
이때 콜백함수로 넘어오는 item은 jQuery 객체가 아닌 javascript의 DOM 객체이므로 jQuery 함수를 사용하고자 한다면, jQuery로 한번 감싸워야 합니다.

<body>
    <ul class="listClass">
        <li>United States of America</li>
        <li>France</li>
        <li>Japan</li>
    </ul>

    <script>
        $('.listClass li').each(function(index, item){
            var txt = $(item).text(); // 콜백함수로 넘어온 item을 jQuery에서 사용할 수 있도록 하는 코드
            console.log(txt);
        })
    </script>

```

```

</body>

// 실행 결과 (console)
// United States of America
// France
// Japan

※ 위 예제를 활용하여 다음과 같은 코드 작성 가능

<body>
    <ul class="listClass">
        <li>후라이드 치킨</li>
        <li>양념 치킨</li>
        <li>훠니콤보</li>
    </ul>

    <script>
        $('.listClass li').each(function(index, item){
            $(item).addClass('li_'+index);           // item에 클래스명을 추가하는 코드
        });

        $('.listClass li').each(function(index, item){
            console.log(item.className);
        })
    </script>
</body>

// 실행 결과
// li_00
// li_01
// li_02

```

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>홀수/짝수 편별</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="wrap">
        <div class="numbox">13</div>
        <div class="numbox">3</div>
        <div class="numbox">14</div>
        <div class="numbox">7</div>
        <div class="numbox">22</div>
        <div class="numbox">38</div>
        <div class="numbox">17</div>
        <div class="numbox">15</div>
        <div class="numbox">11</div>
        <div class="numbox">10</div>
        <div class="numbox">22</div>
        <div class="numbox">31</div>
        <div class="numbox">32</div>
        <div class="numbox">41</div>
        <div class="numbox">10</div>
        <div class="numbox">85</div>
        <div class="numbox">17</div>
        <div class="numbox">82</div>
        <div class="numbox">70</div>
        <div class="numbox">5</div>
        <div class="numbox">53</div>
        <div class="numbox">7</div>
        <div class="numbox">4</div>
        <div class="numbox">29</div>

        <button id="odd-btn">홀수</button>
        <button id="even-btn">짝수</button>
    </div>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3OJU5yExlq6GSYGSHK7tPXikynS7ogEvDejm4=" crossorigin="anonymous"></script>
    <script src="oe.js"></script>
</body>
</html>

```

- style.css

```
@charset "UTF-8";  
  
body{  
    width: 100%;  
    height: 100vh;  
    background-color: #ccc;  
}  
  
.wrap{  
    width: 800px;  
    height: 350px;  
    padding: 20px 0;  
    display: flex;  
    justify-content: center;  
    gap: 20px;  
    flex-wrap: wrap;  
    margin: 0 auto;  
}  
  
.numbox{  
    width: 80px;  
    height: 80px;  
    background-color: #fff;  
    text-align: center;  
    line-height: 80px;  
    font-size: 25px;  
    border-radius: 10px;  
}  
  
button{  
    border: none;  
    background-color: #ccc;  
    color: dodgerblue;  
    font-size: 20px;  
    font-weight: bold;  
    cursor: pointer;  
}  
  
.selected{  
    color: #fff;  
    background-color: #4050B0;  
}
```

- js

```
// 홀수 선택  
$('#odd-btn').click(function(){  
    // 모든 div에 'selected' 클래스 제거  
    $('.numbox').removeClass('selected')  
  
    // each 함수를 이용하여 모든 div의 내용을 가져온다  
    // 가져온 div 내용을 숫자 자료형으로 변환 하여 2로 나눈 나머지가 1일 경우 'selected' 클래스 추가  
    $('.numbox').each(function(){  
        if(Number($(this).text()) % 2 === 1){  
            $(this).addClass('selected')  
        } else{  
            // $(this).removeClass('selected')  
        }  
    })  
})  
  
// 짝수 선택  
$('#even-btn').click(function(){  
    // 모든 div에 'selected' 클래스 제거  
    $('.numbox').removeClass('selected')  
  
    // each 함수를 이용하여 모든 div의 내용을 가져온다  
    // 가져온 div 내용을 숫자 자료형으로 변환 하여 2로 나눈 나머지가 0일 경우 'selected' 클래스 추가  
    $('.numbox').each(function(){  
        if(Number($(this).text()) % 2 === 0){  
            $(this).addClass('selected')  
        } else{  
            // $(this).removeClass('selected')  
        }  
    })  
})
```

- 애니메이션 기본명령

⇒ \$('셀렉터').애니메이션명령(시간의 양)

⇒ 시간의 양은 1초를 1000으로 표기한다

⇒ 시간의 양이 적으면 빨라지고, 많으면 느려진다

⇒ 시간의 양이 움직이는 속도를 나타낸다

⇒ 애니메이션 명령 사용시 시간값을 생략할 수 있다

⇒ 제이쿼리 공식문서에 애니메이션 명령에 대한 시간값이 600으로 입력되어 있어서 시간값을 생략할 수 있다

⇒ \$('셀렉터').애니메이션명령(시간,easing)

- easing : 가속방식(빠르게, 천천히 -> 빠르게, 같은속도 유지, 제이쿼리에서는 기본적으로 2가지 방식을 가지고 있다)

- swing : 점점 빨라진다

- linear : 일정한 속도로 움직인다

⇒ \$('셀렉터').애니메이션명령(시간, easing, 콜백함수)

※ 콜백함수란 ? 애니메이션 완료된 후 실행할 함수

- **show(), hide(), toggle()** : 너비, 높이, 투명도를 변경해서 움직임을 만든다

- **slideDown(), slideUp(), slideToggle()** : 높이값을 해서 움직임을 만든다

- **fadeIn(), fadeOut(), fadeToggle()** : 투명도를 변경해서 움직임을 만든다

※ 위의 9가지 기본명령은 애니메이션이 완료된 후 태그의 display 상태값을 block 또는 none로 만들어준다

⇒ 단, **show(), hide(), toggle()**의 경우 시간값을 생략해서 사용하면 속도값이 0으로 인식된다 (동작 발생 즉시 원상태로 바뀐다는 이야기)

※ 애니메이션 예시

- **index.html**

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>09 애니메이션 기본명령</title>
    <link rel="stylesheet" href="css/style09.css">
</head>
<body>
    <h1>애니메이션 기본명령</h1>
    <div class="box1">
        <div class="bth">
            <button class="bt1">show</button>
            <button class="bt2">hide</button>
            <button class="bt3">toggle</button>
            <button class="bt4">slideDown</button>
            <button class="bt5">slideUp</button>
            <button class="bt6">slideToggle</button>
            <button class="bt7">fadeIn</button>
            <button class="bt8">fadeOut</button>
            <button class="bt9">fadeToggle</button>
        </div>
        <div class="pan"></div>
    </div>
    <br><br>

    <div class="box2">
        <button>CLICK</button>
        <div class="b2">
            <p class="b21">swing</p>
            <p class="b22">linear</p>
        </div>
    </div>
    <br><hr><br>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="js/s09.js"></script>
</body>
</html>
```

- **style.css**

```
@charset "utf-8";  
  
.box1{  
    display: flex;  
    height: 300px;  
    margin-top: 50px;  
}  
  
.box1 .btn{  
    width: 300px;  
}  
  
.box1 .btn button{  
    width: 80px;  
    height: 30px;  
    margin-bottom: 10px;  
}  
  
.box1 .pan{  
    width: 300px;  
    height: 300px;  
    margin: 0;  
    background-color: salmon;  
}  
  
.box2{  
    margin: 30px 0;  
}  
  
.box2 .b2{  
    display: flex;  
    height: 500px;  
}  
  
.box2 .b2 p{  
    width: 200px;  
    height: 500px;  
    margin-right: 20px;  
    background-color: violet;  
}
```

- **common.js**

```
$('.bt1').click(function(){  
    $('.box .pan').show(5000)  
})  
  
$('.bt2').click(function(){  
    $('.box1 .pan').hide(5000)  
})  
  
$('.bt3').click(function(){  
    $('.box1 .pan').toggle(2000)  
})  
  
$('.bt4').click(function(){  
    $('.box1 .pan').slideDown(2000)  
})  
  
$('.bt5').click(function(){  
    $('.box1 .pan').slideUp(2000)  
})  
  
$('.bt6').click(function(){  
    $('.box1 .pan').slideToggle(2000)  
})  
  
$('.bt7').click(function(){  
    $('.box1 .pan').fadeIn(2000)  
})  
  
$('.bt8').click(function(){  
    $('.box1 .pan').fadeOut(2000)  
})  
  
$('.bt9').click(function(){  
    $('.box1 .pan').fadeToggle(2000)  
})
```

```

$('.box2 button').click(function(){
    $('.box2 .b21').slideToggle(5000)
    $('.box2 .b22').slideToggle(5000, 'linear')
})

```

- **animate 명령**

- ⇒ 스타일 속성값이 변경되는 것을 움직임으로 만들어주는 명령
- ⇒ 값을 숫자로 표기할 수 있는 속성만 적용가능하다. 오로지 숫자로 되는 값만 사용 가능하다
- ※ **margin, padding 등. transform:scale 이런것은 불가능하다**
- ⇒ \$('셀렉터').animate(스타일, 시간, easing, 클백함수)
- ⇒ \$('셀렉터').animate({스타일 속성: '값'}, 시간, easing, 클백함수)

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>애니메이션</title>
    <style>
        div{
            background-color: #2dbaf2;
            position: absolute;
            height: 100px;
            width: 100px;
        }
    </style>
</head>
<body>
    <button>애니메이션</button>
    <div></div>

    <button class="go-to-top">Top</button>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3OJU5yExlq6GSYGSHK7tPXikynS7ogEvDej/m4=" crossorigin=>
    <script>
        $(document).ready(function(){
            $('button').on('click', function(){
                $('div').animate({
                    left: '250px', opacity: '0.5'
                })
            })
        })
    </script>
</body>
</html>

```

- **stop과 delay 명령**

- ⇒ 이벤트가 연속해서 발생하게 되면 지정된 애니메이션도 이벤트의 발생 횟수만큼 반복해서 실행하게 된다
- ⇒ 더이상 애니메이션이 필요없을 때도 대상이 계속 움직이게 된다
- ⇒ 그래서 애니메이션 명령 앞에 stop() 명령을 추가한다
- ⇒ stop() 명령이 적용되면, 애니메이션 명령이 즉시 멈추게 되고, 움직임을 한번만 실행시켜준다
- ⇒ delay() 명령으로 애니메이션이 시작되는 시점을 늦출 수 있다
- ⇒ \$('셀렉터').delay(시간).애니메이션명령()

※ **animate 명령 예시**

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<title>10 animate 명령</title>
<link rel="stylesheet" href="css/style10.css">
</head>
<body>
  <div class="box1">
    <button>CLICK</button>
    <p></p>
  </div>

  <div class="box2">
    <p></p>
  </div>

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
  <script src="js/s10.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "utf-8";

.box1 p{
  width: 200px;
  height: 200px;
  background-color: lime;
}

.box2{
  height: 300px;
  padding: 30px;
  background-color: antiquewhite;
}

.box2 p{
  height: 300px;
  margin: 0;
  background-color: aquamarine;
}

```

- **common.js**

```

$('.box1 button').click(function(){
  $('.box1 p').animate({
    marginLeft: 300,
    marginTop: 300,
    width:400
  }, 2000, function(){
    $(this).css({
      backgroundColor: "red"
    })
  })
})

$('.box2').mouseover(function(){
  $('.box2 p').delay(2000).stop().fadeOut()
}).mouseout(function(){
  $('.box2 p').delay(2000).stop().fadeIn()
})

```

[JQuery 예제]

1. 갤러리 페이지 (참고 : <https://5603.tistory.com/154>) ※ tab 클릭 시 이미지 변경

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>갤러리 페이지</title>
  <link rel="stylesheet" href="css/ex01.css">
</head>
<body>
  <div class="box">
    <div class="photo">

```

```

<ul>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
</ul>
</div>
<div class="btn">
    <ul>
        <li class="on">1</li>
        <li>2</li>
        <li>3</li>
        <li>4</li>
        <li>5</li>
    </ul>
</div>
</div>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="js/ex01.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "utf-8";

ul{
    list-style: none;
    padding: 0;
}

.box{
    width: 300px;
    margin: 0 auto;
}

.box .photo li{
    display: none;
}

.box .photo li:first-child{
    display: block;
}

.box .btn ul{
    display: flex;
    justify-content: center;
}

.box .btn li{
    width: 15px;
    height: 15px;
    margin: 0 5px;
    background-color: #ccc;
    /* 텍스트를 요소의 가장 뒤로 보내는 속성 */
    text-indent: -99999px;
    border-radius: 50%;
    cursor: pointer;
}

.box .btn .on{
    background-color: lightblue;
}

```

- **Ex01.js**

```

/*
기본 Logic
1. 버튼 클릭 이벤트
1) 클릭한 버튼의 순서값 저장
2) 버튼 전체 비활성화
3) 클릭한 버튼 활성화
4) 사진 전체 비활성화
5) 클릭한 버튼과 동일한 순서값을 가진 사진 활성화
*/
$('.btn li').click(function(){
    let i = $(this).index()
    console.log(i)
})

```

```

$('.btn li').removeClass('on')
$(this).addClass('on')
$('.photo li').css({
    display : "none"
})
$('.photo li').eq(i).css({
    display : "block"
})
})
})

```

※ Escape sequence (HTML)

```

&      =>      &amp;
<      =>      &lt;
=      =>      -
>      =>      &gt;
=>      &nbsp;
""     =>      &quot;
'''    =>      &#39;
/      =>      &#x2F;
`      =>      &#x60;
=      =>      &#x3D;

```

- 캐러셀

⇒ Carousel이란 여러 개의 이미지나 영상을 슬라이드 형태로 만들어서 버튼을 누를 때마다

이미지가 바뀌는 것을 의미합니다.

- index.html

```

<section class="carouselContainer">
  <div class="carousel">
    <div class="carousel_item">
      
    </div>

    <div class="carousel_item">
      
    </div>

    <div class="carousel_item">
      
    </div>

    <div class="carousel_item">
      
    </div>

    <div class="carousel_item">
      
    </div>
  </div>
</section>

```

- style.css

```

.carouselContainer {
  width: 500px; //현재 보여줄 구역의 width
  height: 300px;
  overflow-x: hidden; // 현재 구역을 제외한 부분 hidden처리
  overflow-y: hidden;
  margin: auto;
}

.carouselContainer > .carousel {
  display: flex; // 이미지를 x축으로 정렬
  transform: translate3d(0, 0, 0); // 초기에는 0위치에서 시작함
  transition: transform 0.2s; // 부드러운 애니메이션을 위함
}

.carousel_item {
  width: 500px; // 이미지 크기를 맞춰주기위한 width
  height: 300px;
}

```

```
.carousel_item > img {
  width: 500px;
  height: 300px;
  object-fit: contain; // 실제 이미지를 상위 div에 딱 맞춰주기위함
}
```

- js

```
// prev 버튼을 눌렀을때 실행
prev() {
  if (this.index === 0) return;
  this.index -= 1;

  this.$carousel.style.transform = `translate3d(-${
    500 * this.index
  }px, 0, 0)`;

}

// next 버튼을 눌렀을때 실행
next() {
  if (this.index === 4) return;
  this.index += 1;

  this.$carousel.style.transform = `translate3d(-${
    500 * this.index
  }px, 0, 0)`;

}
```

```
class Carousel {
  index = 0;
  catList;

  constructor({ $target, initialData }) {
    const $prevButton = document.createElement("button");
    const $nextButton = document.createElement("button");
    const $buttonContainer = document.createElement("section");

    const $carouselContainer = document.createElement("section");
    const $carousel = document.createElement("div");

    $prevButton.className = "prevButton";
    $nextButton.className = "nextButton";
    $buttonContainer.classList = "buttonContainer";

    $carouselContainer.className = "carouselContainer";
    $carousel.classList = "carousel";

    this.$prevButton = $prevButton;
    this.$nextButton = $nextButton;
    this.$buttonContainer = $buttonContainer;

    this.$carouselContainer = $carouselContainer;
    this.$carousel = $carousel;

    this.$prevButton.innerText = "Prev";
    this.$nextButton.innerText = "Next";

    this.$prevButton.addEventListener("click", () => {
      this.prev();
    });

    this.$nextButton.addEventListener("click", () => {
      this.next();
    });

    this.$buttonContainer.appendChild(this.$prevButton);
    this.$buttonContainer.appendChild(this.$nextButton);
    this.$carouselContainer.appendChild(this.$carousel);

    $target.appendChild(this.$carouselContainer);
    $target.appendChild(this.$buttonContainer);

    this.get50Cats();
  }

  // 고양이 50개 이미지를 받아온다음 5개만 가져오는 api 함수
  async get50Cats() {
    try {
      const { data } = await api.fetchCat50();
      this.catList = data.slice(0, 5);
    }
  }
}
```

```

        this.render();
    } catch (e) {
        console.log(e);
    }
}

prev() {
    if (this.index === 0) return;
    this.index -= 1;

    this.$carousel.style.transform = `translate3d(-${
        500 * this.index
    }px, 0, 0)`;;
}

next() {
    if (this.index === 4) return;
    this.index += 1;

    this.$carousel.style.transform = `translate3d(-${
        500 * this.index
    }px, 0, 0)`;;
}

render() {
    this.$carousel.innerHTML = this.catList
        .map((cat, index) => {
            return `
                <div class='carousel_item'>
                    <img src=${cat.url} alt=${cat.name} />
                </div>
            `;
        })
        .join("");
}
}

```

2. PC 기본메뉴 #1 : 마우스 오버 시 sub 메뉴 전체 슬라이드 다운 효과

슬라이드 기능 구현 참고 : <https://thestorybook.tistory.com/134>

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PC 기본메뉴 1 - 마우스 오버 시 sub 메뉴 전체 슬라이드 다운</title>
    <link rel="stylesheet" href="css/ex02.css">
</head>
<body>
    <div id="wrap">
        <header id="header" >
            <div class="h_in">
                <nav class="gnb">
                    <ul class="menu">
                        <li class="d1">
                            <a class="main" href="#">리바트브랜드</a>
                            <div class="sub">
                                <ul>
                                    <li><a href="#">리바트</a></li>
                                    <li><a href="#">리바트 이즈마인</a></li>
                                    <li><a href="#">리바트 키즈</a></li>
                                    <li><a href="#">H.MONDO</a></li>
                                    <li><a href="#">리챈</a></li>
                                    <li><a href="#">리바트 키친</a></li>
                                    <li><a href="#">리바트 하움</a></li>
                                    <li><a href="#">리바트 네오스</a></li>
                                    <li><a href="#">리바트 빌트인</a></li>
                                    <li><a href="#">리바트 마린</a></li>
                                    <li><a href="#">리바트 앤솔림</a></li>
                                    <li><a href="#">법인영업(H&S)</a></li>
                                </ul>
                            </div>
                        </li>
                        <li class="d1">
                            <a class="main" href="#">리바트몰</a>
                            <div class="sub">
                                <ul>
                                    <li><a href="#">기획전</a></li>
                                    <li><a href="#">이벤트</a></li>
                                </ul>
                            </div>
                        </li>
                    </ul>
                </nav>
            </div>
        </header>
        <div class="contents" style="background-color: #f0f0f0; height: 400px;">
```

```

        <li><a href="#">빈짝딜</a></li>
        <li><a href="#">HOT 50</a></li>
        <li><a href="#">테마존</a></li>
        <li><a href="#">패키지 상품</a></li>
        <li><a href="#">아울렛</a></li>
    </ul>
</div>
</li>
<li class="d1">
    <a class="main" href="#">리바트스토리</a>
    <div class="sub">
        <ul>
            <li><a href="#">리바트 미디어</a></li>
            <li><a href="#">리바트 신제품</a></li>
            <li><a href="#">리바트 다이어리</a></li>
            <li><a href="#">브랜드 이벤트</a></li>
            <li><a href="#">매장 이벤트</a></li>
            <li><a href="#">주방시공사례</a></li>
        </ul>
    </div>
</li>
<li class="d1">
    <a class="main" href="#">회사소개</a>
    <div class="sub">
        <ul>
            <li><a href="#">About 현대리바트</a></li>
            <li><a href="#">연구소</a></li>
            <li><a href="#">투자정보</a></li>
            <li><a href="#">인재채용</a></li>
            <li><a href="#">고객센터</a></li>
            <li><a href="#">매장찾기</a></li>
        </ul>
    </div>
</li>
</ul>
</nav>
</div>
<div class="gnbBg"></div>
</header>
</div>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="js/ex02.js"></script>
</body>
</html>

```

• style.css

```

@charset "utf-8";

body{
    margin: 0;
}

ul{
    list-style: none;
    padding: 0;
    margin: 0;
}

a{
    text-decoration: none;
    color: initial;
}

/* reset */

#wrap{
    position: relative;
    width: 100%;
    min-width: 1220px;
}

#header{
    /* 상단에 위치만 고정하고 싶을 때 사용, fixed의 경우 스크롤에 움직임에 따라 오도록 할때 */
    position: relative;
    border-bottom: 1px solid grey;
    z-index: 500;
}

.h_in{
    position: relative;
    width: 1200px;
    height: 80px;
}

```

```

margin: 0 auto;
background-color: aliceblue;
z-index: 100;
}

.gnb{
  position: absolute;
  top: 40px;
  right: 0;
  height: 40px;
  overflow: hidden;
}

.menu{
  display: flex;
}

.menu .d1{
  position: relative;
}

.menu .d1 .main{
  display: block;
  /* padding : 위 오른 아래 원 */
  padding: 0 50px 40px 50px;
  font-size: 17px;
  font-weight: 700;
  background-color: cyan;
}

.gnb .sub{
  /* 위치변동 없이 단순 고정 목적으로 사용할 경우 position 사용 */
  position: absolute;
  top: 40px;
  left: 0;
  width: 100%;
  background-color: #eee;
}

.gnb .sub ul{
  padding: 10px 0;
  text-align: center;
}

.gnb .sub li{
  padding: 4px 0;
}

.gnb .sub li a{
  font-size: 15px;
  color: #444;
}

.gnbBg{
  position: absolute;
  top: 80px;
  left: 0;
  display: none;
  width: 100%;
  height: 385px;
  background-color: lightblue;
}

```

- **Ex02-1.js**

```

1) 애니메이션이 없는 움직임
$('.gnb').mouseover(function(){
  $(this).css({
    height : 410
  })
  // show() = display:none , hide() = display:block
  // hover와 같은 효과는 stop()을 사용하지 않아도 된다
  $('.gnbBg').show()
}).mouseout(function(){
  $(this).css({
    // gnb의 높이값만큼은 남겨둬야 함, height : 0으로 할 경우 전체 요소가 사라짐
    height : 40
  })
  $('.gnbBg').hide()
})

2) 애니메이션이 있는 움직임
$('.gnb').mouseover(function(){
  $(this).stop().animate({

```

```

        height : 410
    })
    $('.gnbBg').stop().slideDown()
}).mouseout(function(){
    $(this).stop().animate({
        height : 40
    })
    $('.gnbBg').stop().slideUp()
})

```

2. PC 기본메뉴 #2 : 마우스 오버 시 선택하는 메뉴의 sub 메뉴 슬라이드 다운

- **find()** : 셀렉터 영역 안쪽에 작성된 태그 중 원하는 대상을 선택한다

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PC 기본메뉴 2 - 마우스 오버 시 선택하는 메뉴의 sub 메뉴 슬라이드 다운</title>
    <link rel="stylesheet" href="css/ex03.css">
</head>
<body>
    <div id="wrap">
        <header id="header" >
            <div class="h_in">
                <nav class="gnb">
                    <ul class="menu">
                        <li class="d1">
                            <a class="main" href="#">리바트브랜드</a>
                            <div class="sub">
                                <ul>
                                    <li><a href="#">리바트</a></li>
                                    <li><a href="#">리바트 이즈마인</a></li>
                                    <li><a href="#">리바트 키즈</a></li>
                                    <li><a href="#">H_MONDO</a></li>
                                    <li><a href="#">리챈</a></li>
                                    <li><a href="#">리바트 키친</a></li>
                                    <li><a href="#">리바트 하음</a></li>
                                    <li><a href="#">리바트 네오스</a></li>
                                    <li><a href="#">리바트 빌트인</a></li>
                                    <li><a href="#">리바트 마린</a></li>
                                    <li><a href="#">리바트 앤솔림</a></li>
                                    <li><a href="#">법인영업(H&S)</a></li>
                                </ul>
                            </div>
                        </li>
                        <li class="d1">
                            <a class="main" href="#">리바트몰</a>
                            <div class="sub">
                                <ul>
                                    <li><a href="#">기획전</a></li>
                                    <li><a href="#">이벤트</a></li>
                                    <li><a href="#">반짝딜</a></li>
                                    <li><a href="#">HOT 50</a></li>
                                    <li><a href="#">테마존</a></li>
                                    <li><a href="#">패키지 상품</a></li>
                                    <li><a href="#">아울렛</a></li>
                                </ul>
                            </div>
                        </li>
                        <li class="d1">
                            <a class="main" href="#">리바트스토리</a>
                            <div class="sub">
                                <ul>
                                    <li><a href="#">리바트 미디어</a></li>
                                    <li><a href="#">리바트 신제품</a></li>
                                    <li><a href="#">리바트 다이어리</a></li>
                                    <li><a href="#">브랜드 이벤트</a></li>
                                    <li><a href="#">매장 이벤트</a></li>
                                    <li><a href="#">주방시공사례</a></li>
                                </ul>
                            </div>
                        </li>
                        <li class="d1">
                            <a class="main" href="#">회사소개</a>
                            <div class="sub">
                                <ul>

```

```

        <li><a href="#">About 현대리바트</a></li>
        <li><a href="#">연구소</a></li>
        <li><a href="#">투자정보</a></li>
        <li><a href="#">인재채용</a></li>
        <li><a href="#">고객센터</a></li>
        <li><a href="#">매장찾기</a></li>
    </ul>
</div>
</ul>
</nav>
</div>
</header>
</div>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="js/ex03.js"></script>
</body>
</html>

```

• style.css

```

@charset "utf-8";

body{
    margin: 0;
}
ul{
    list-style: none;
    padding: 0;
    margin: 0;
}

/* reset */

#wrap{
    position: relative;
    width: 100%;
    min-width: 1220px;
}

#header{
    position: relative;
    border-bottom: 1px solid #ccc;
    z-index: 500;
}

.h_in{
    position: relative;
    width: 1200px;
    height: 80px;
    margin: 0 auto;
    background-color: aliceblue;
    z-index: 100;
}

.gnb{
    position: absolute;
    top: 40px;
    right: 0;
}

.menu{
    display: flex;
}

.menu .d1{
    position: relative;
}

.menu .d1 .main{
    display: block;
    padding: 0 50px 20px 50px;
    font-size: 17px;
    font-weight: 700;
    text-decoration: none;
    color: #222;
    background-color: cyan;
}

.gnb .sub{
    position: absolute;
    top: 40px;

```

```

    left: 0;
    display: none;
    width: 100%;
    background-color: #eee;
}

.gnb .sub ul{
    padding: 10px 0;
    text-align: center;
}

.gnb .sub li{
    padding: 4px 0;
}

.gnb .sub li a{
    font-size: 15px;
    color: #444;
    text-decoration: none;
}

```

- Ex02-2.js

```

$('.gnb .d1').mouseover(function(){
    // 마우스 오버시 상단 메뉴에 모든 요소 비활성화
    $('.gnb .d1 .sub').hide()

    // 마우스가 올려지는 대상만 보여지도록 즉, sub 클래스의 부모요소
    // gnb(최상단 메뉴를 감싸는 wrap)와 d1(최상단 각각의 메뉴) 의 안쪽 요소 중 sub(sub 메뉴 전체) 선택
    $(this).find('.sub').show()
}).mouseout(function(){
    // 현재 마우스가 올려져 있는 대상의 sub 메뉴 숨겨지도록
    $(this).find('.sub').hide()
    // $('.gnb .d1 .sub').hide()
})

```

2. PC 기본메뉴 #3 : 마우스 오버 시 선택하는 메뉴의 sub 메뉴 슬라이드 다운(sub 메뉴 영역은 전체를 차지하도록)

- **height()** : 셀렉터의 높이값을 저장한다.(여백을 제외하고 실제 컨텐츠가 적용되는 영역을 가리킨다)
- **innerHeight()** : 내부높이 + padding
- **outerHeight()** : 내부높이 + padding + border
- **outerHeight(true)** : 내부높이 + padding + border + margin
- **width()** : 같은 느낌의 너비값을 저장한다.(여백을 제외하고 실제 컨텐츠가 적용되는 영역을 가리킨다)

- index.html

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PC 기본메뉴 3 - 마우스 오버 시 선택하는 메뉴의 sub 메뉴 슬라이드 다운(sub 메뉴는 영역의 높이값과 상단의 메뉴의 너비값만큼 차지)</title>
    <link rel="stylesheet" href="css/ex04.css">
</head>
<body>
    <div id="wrap">
        <header id="header" >
            <div class="h_in">
                <nav class="gnb">
                    <ul class="menu">
                        <li class="d1">
                            <a class="main" href="#">리바트브랜드</a>
                            <div class="sub">
                                <ul>
                                    <li><a href="#">리바트</a></li>
                                    <li><a href="#">리바트 이즈마인</a></li>
                                    <li><a href="#">리바트 키즈</a></li>
                                    <li><a href="#">H.MONDO</a></li>
                                    <li><a href="#">리챈</a></li>
                                    <li><a href="#">리바트 키친</a></li>
                                    <li><a href="#">리바트 하음</a></li>
                                    <li><a href="#">리바트 네오스</a></li>
                                    <li><a href="#">리바트 빌트인</a></li>
                                </ul>
                            </div>
                        </li>
                    </ul>
                </nav>
            </div>
        </header>
    </div>

```

```

        <li><a href="#">리바트 마린</a></li>
        <li><a href="#">리바트 앤슬립</a></li>
        <li><a href="#">법인영업(H&S)</a></li>
    </ul>
</div>
</li>
<li class="d1">
    <a class="main" href="#">리바트몰</a>
    <div class="sub">
        <ul>
            <li><a href="#">기획전</a></li>
            <li><a href="#">이벤트</a></li>
            <li><a href="#">반찌딜</a></li>
            <li><a href="#">HOT 50</a></li>
            <li><a href="#">테마존</a></li>
            <li><a href="#">패키지 상품</a></li>
            <li><a href="#">아울렛</a></li>
        </ul>
    </div>
</li>
<li class="d1">
    <a class="main" href="#">리바트스토리</a>
    <div class="sub">
        <ul>
            <li><a href="#">리바트 미디어</a></li>
            <li><a href="#">리바트 신제품</a></li>
            <li><a href="#">리바트 다이어리</a></li>
            <li><a href="#">브랜드 이벤트</a></li>
            <li><a href="#">매장 이벤트</a></li>
            <li><a href="#">주방시공사례</a></li>
        </ul>
    </div>
</li>
<li class="d1">
    <a class="main" href="#">회사소개</a>
    <div class="sub">
        <ul>
            <li><a href="#">About 현대리바트</a></li>
            <li><a href="#">연구소</a></li>
            <li><a href="#">투자정보</a></li>
            <li><a href="#">인재채용</a></li>
            <li><a href="#">고객센터</a></li>
            <li><a href="#">매장찾기</a></li>
        </ul>
    </div>
</li>
</ul>
</nav>
</div>
<div class="gnbBg"></div>
</header>
</div>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="js/ex04.js"></script>
</body>
</html>

```

• style.css

```

@charset "utf-8";

body{
    margin: 0;
}

ul{
    list-style: none;
    padding: 0;
    margin: 0;
}
/* reset */

#wrap{
    position: relative;
    width: 100%;
    min-width: 1220px;
}

#header{
    position: relative;
    border-bottom: 1px solid #ccc;
    z-index: 500;
}

```

```

.h_in{
    position: relative;
    width: 1200px;
    height: 80px;
    margin: 0 auto;
    background-color: aliceblue;
    z-index: 100;
}

.gnb{
    position: absolute;
    top: 40px;
    right: 0;
}

.menu{
    display: flex;
}

.menu .d1 .main{
    display: block;
    padding: 0 50px 20px 50px;
    font-size: 17px;
    font-weight: 700;
    text-decoration: none;
    color: #222;
    background-color: cyan;
}

.gnb .sub{
    position: absolute;
    top: 40px;
    left: 0;
    display: none;
    width: 100%;
    background-color: #eee;
}

.gnb .sub ul{
    padding: 10px 0;
}

.gnb .sub li{
    padding: 4px 0;
}

.gnb .sub li a{
    font-size: 15px;
    color: #444;
    text-decoration: none;
}

.gnbBg{
    position: absolute;
    top: 80px;
    left: 0;
    display: none;
    width: 100%;
    height: 385px;
    background-color: lightblue;
}

```

- **Ex02-3.js**

```

$('.gnb .d1').mouseover(function(){
    let h = $(this).find('.sub').height()
    $('.gnb .d1 .sub').hide()
    $(this).find('.sub').show()
    // $('.gnbBg').css({height : h}) 구문을 생략해도 무관함 (단, 해당 구문을 생략할 경우 컨텐츠만큼 높이값이 지정되는 것이 아니라 css에서 지정한 높이값으로 고정)
    $('.gnbBg').css({
        height : h
    })
    $('.gnbBg').show()
}).mouseout(function(){
    $(this).find('.sub').hide()
    $('.gnbBg').hide()
})

```

※ 메뉴 과제 풀이

- CSS 애니메이션 예제 : <https://ldrerin.tistory.com/397>

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>메뉴 과제 - css만으로 만들기</title>
    <link rel="stylesheet" href="menu.css">
</head>
<body>
    <div id="wrap">
        <header id="header" >
            <div class="h_in">
                <nav class="gnb">
                    <ul class="menu">
                        <li class="d1">
                            <a class="main" href="#">리바트브랜드</a>
                            <div class="sub">
                                <ul>
                                    <li><a href="#">리바트</a></li>
                                    <li><a href="#">리바트 이즈마인</a></li>
                                    <li><a href="#">리바트 키즈</a></li>
                                    <li><a href="#">H.MONDO</a></li>
                                    <li><a href="#">리천</a></li>
                                    <li><a href="#">리바트 키친</a></li>
                                    <li><a href="#">리바트 하옴</a></li>
                                    <li><a href="#">리바트 네오스</a></li>
                                    <li><a href="#">리바트 빌트인</a></li>
                                    <li><a href="#">리바트 마린</a></li>
                                    <li><a href="#">리바트 앤솔립</a></li>
                                    <li><a href="#">법인영업(H&S)</a></li>
                                </ul>
                            </div>
                        </li>
                        <li class="d1">
                            <a class="main" href="#">리바트몰</a>
                            <div class="sub">
                                <ul>
                                    <li><a href="#">기획전</a></li>
                                    <li><a href="#">이벤트</a></li>
                                    <li><a href="#">반짝딜</a></li>
                                    <li><a href="#">HOT 50</a></li>
                                    <li><a href="#">테마존</a></li>
                                    <li><a href="#">패키지 상품</a></li>
                                    <li><a href="#">아울렛</a></li>
                                </ul>
                            </div>
                        </li>
                        <li class="d1">
                            <a class="main" href="#">리바트스토리</a>
                            <div class="sub">
                                <ul>
                                    <li><a href="#">리바트 미디어</a></li>
                                    <li><a href="#">리바트 신제품</a></li>
                                    <li><a href="#">리바트 다이어리</a></li>
                                    <li><a href="#">브랜드 이벤트</a></li>
                                    <li><a href="#">매장 이벤트</a></li>
                                    <li><a href="#">주방시공사례</a></li>
                                </ul>
                            </div>
                        </li>
                        <li class="d1">
                            <a class="main" href="#">회사소개</a>
                            <div class="sub">
                                <ul>
                                    <li><a href="#">About 현대리바트</a></li>
                                    <li><a href="#">연구소</a></li>
                                    <li><a href="#">투자정보</a></li>
                                    <li><a href="#">인재채용</a></li>
                                    <li><a href="#">고객센터</a></li>
                                    <li><a href="#">매장찾기</a></li>
                                </ul>
                            </div>
                        </li>
                    </ul>
                </nav>
            </div>
        <!-- <div class="gnbBg"></div> -->
    </header>

```

```
</div>
</body>
</html>
```

- **style.css**

```
@charset "utf-8";

body {
    margin: 0;
}

ul {
    list-style: none;
    padding: 0;
    margin: 0;
}

a{
    text-decoration: none;
    color: initial;
}
/* reset */

#wrap {
    position: relative;
    width: 100%;
    min-width: 1220px;
}

#header {
    position: relative;
    border-bottom: 1px solid #ccc;
    z-index: 500;
}

.h_in {
    position: relative;
    width: 1200px;
    height: 80px;
    margin: 0 auto;
    background-color: aliceblue;
    z-index: 100;
}

.gnb {
    position: absolute;
    top: 40px;
    right: 0;
}

.menu {
    display: flex;
}

.menu:hover .d1 .sub{
    height: 400px;
}

.menu .d1{
    position: relative;
}

.menu .d1 .main {
    display: block;
    padding: 0 50px 20px 50px;
    font-size: 17px;
    font-weight: 700;
    background-color: cyan;
}

.gnb .sub {
    position: absolute;
    top: 40px;
    left: 0;
    width: 100%;
    background-color: #eee;
    height: 0;
    overflow: hidden;
    transition: all 3s;
}

.gnb .sub ul {
```

```

        padding: 10px 0;
        text-align: center;
    }

    .gnb .sub li {
        padding: 4px 0;
    }

    .gnb .sub li a {
        font-size: 15px;
        color: #444;
    }

    /* .gnbBg {
        position: absolute;
        top: 80px;
        left: 0;
        width: 100%;
        height: 385px;
        background-color: lightblue;
    } */

    /* 마우스 hover 효과 */
    /* .gnb:hover .sub{
        animation: slideDown 2s linear infinite alternate;
    }

    @keyframes slideDown {
        to{
            height: 380px;
        }
    }

    .gnb:not(:hover) .sub{
        transition: all 2s ease-out;
    } */

    /* gnbBg의 경우 gnb의 상위 요소 즉, inner(h_in)의 형제요소임
    상위선택자에게 효과를 적용할 수 없음
    */
    /* .gnb:hover .gnbBg{
    } */

```

2. PC 기본 메뉴 #4 - SlideMenu (우 → 좌측으로 나타나는 메뉴)

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ex01. SlideMenu</title>
    <link rel="stylesheet" href="css/reset.css">
    <link rel="stylesheet" href="css/Ex01.css">
    <!-- <link rel="stylesheet" href="css/style.css"> -->
</head>
<body>
    <div id="wrap">
        <header>
            <h1>LOGO</h1>
            <div class="btn_m">
                <span></span>
                <span></span>
                <span></span>
            </div>
            <!-- btn_m_end -->

            <p class="bkbg"></p>

            <nav>
                <p class="close_sideM">
                    <span></span>
                    <span></span>
                </p>
                <!-- close_sideM_end -->

                <ul class="d1">
                    <li><a href="#">메뉴1</a></li>
                    <li><a href="#">메뉴2</a></li>

```

```

        <li><a href="#">메뉴3</a></li>
        <li><a href="#">메뉴4</a></li>
        <li><a href="#">메뉴5</a></li>
    </ul>
</nav>
<!-- nav_end -->
</header>
</div>
<!-- wrap_end -->
<script src="js/jquery-3.7.0.min.js"></script>
<script src="js/Ex01.js"></script>
</body>
</html>

```

- **style.css #1**

```

@charset "utf-8";

#wrap{
    width: 800px;
    height: 600px;
    background-color: #ccc;
    margin: 20px auto;
    position: relative;
    overflow: hidden;
}
/* wrap_end */

header{
    width: 100%;
    height: 100px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    background-color: #fff;
    border-bottom: 1px solid #ddd;
    box-shadow: 0 2px 8px rgba(0,0,0,0.2)
}

header > h1{
    width: 200px;
    height: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
}

header > .btn_m{
    width: 50px;
    height: 50px;
    background-color: salmon;
    position: relative;
    margin-right: 20px;
    cursor: pointer;
}

.btn_m > span{
    /* span 태그의 경우 position 속성을 적용하면 display 속성이 block로 변경된다 */
    position: absolute;
    top: 14px;
    left: 10px;
    width: 30px;
    height: 3px;
    background-color: #fff;
}

.btn_m > span:nth-child(2){
    top: 50%;
}

.btn_m > span:nth-child(3){
    top: 35px;
}
/* btn_m_end */

.bkbg{
    /* display: none; */
    position: absolute;
    top: 0;
    left: -500px;
    width: 500px;
    height: 100%;
    background-color: rgba(0,0,0,0.8);
}

```

```

        z-index: 100;
    }
/* bkg_end */

nav{
    width: 300px;
    height: 100%;
    background-color: #eee;
    position: absolute;
    top: 0;
    right: -300px;
    padding: 20px;
    z-index: 90;
}

nav > .close_sideM{
    width: 30px;
    height: 30px;
    background-color: green;
    border-radius: 50%;
    position: relative;
    float: right;
    margin-top: 100px;
    cursor: pointer;
}

.close_sideM > span{
    width: 20px;
    height: 3px;
    background-color: #fff;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%,-50%) rotate(45deg);
}

.close_sideM > span:nth-child(2){
    transform: translate(-50%,-50%) rotate(-45deg);
}

.d1{
    width: 100%;
    height: 100%;
    margin-top: 180px;
}

.d1 > li{
    width: 100%;
    height: 40px;
    border-bottom: 1px solid #ccc;
    line-height: 40px;
    padding-left: 10px;
}
/* nav_end */

```

- **style.css #2 (풀이)**

```

@charset "UTF-8";

header{
    display: flex;
    justify-content: space-between;
    align-items: center;
    width: 100%;
    height: 80px;
    padding: 0 50px;
    background-color: lightblue;
    box-shadow: 0 2px 8px rgba(0,0,0,0.2);
}

header .btn_m{
    position: relative;
    width: 50px;
    height: 50px;
    background-color: lightcoral;
    cursor: pointer;
}

header .btn_m span{
    position: absolute;
    left: 10px;
    width: 30px;
    height: 3px;
    background-color: #fff;

```

```

}

header .btn_m span:nth-child(1){
    top: 12px;
}

header .btn_m span:nth-child(2){
    top: 50%;
    margin-top: -1.5px;
}

header .btn_m span:nth-child(3){
    bottom: 12px;
}

header .bkbg{
    display: none;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100vh;
    background-color: rgba(0,0,0,0.6);
}

header nav{
    position: fixed;
    right: -250px;
    top: 0;
    padding: 80px 20px;
    width: 250px;
    height: 100vh;
    background-color: #eaeaea;
}

header nav .close_sideM{
    position: relative;
    float: right;
    width: 30px;
    height: 30px;
    border-radius: 50%;
    background-color: lightgreen;
}

header nav .close_sideM span{
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%) rotate(45deg);
    width: 2px;
    height: 20px;
    background-color: #fff;
}

header nav .close_sideM span:nth-child(2){
    transform: translate(-50%, -50%) rotate(-45deg);
}

header nav .d1{
    margin-top: 60px;
}

header nav .d1 li{
    line-height: 40px;
    border-bottom: 1px solid #ccc;
}

```

- **Ex02-4.js #1**

```

$(document).ready(function(){
    // 메뉴버튼 클릭 시 사이드메뉴 + 백그라운드 배경 나타남
    $('.btn_m').click(function(){
        $('nav').animate({right: 0}, 300)
        $('.bkbg').animate({left: 0}, 300)
    })

    // 닫기버튼 클릭 시 사이드메뉴 + 백그라운드 배경 사라짐
    $('.close_sideM').click(function(){
        $('nav').animate({right: -300}, 300)
        $('.bkbg').animate({left: -500}, 300)
    })
})

```

- Ex02-4.js #2 (풀이)

```

$(document).ready(function(){
    // 메뉴버튼을 누르면 사이드메뉴 + 검정배경 나타남
    $('.btn_m').click(function(){
        $('nav').animate({right:'0'}, 300)
        $('.bkbg').fadeIn()
    })

    // 닫기버튼을 누르면 사이드메뉴 + 검정배경 사라짐
    $('.close_sideM').click(function(){
        $('nav').animate({right:'-250px'}, 200)
        $('.bkbg').fadeOut()
    })

    // 검정배경 누르면 사이드메뉴 + 검정배경 사라짐
    $('.bkbg').click(function(){
        $('nav').animate({right:'-250px'}, 100)
        $(this).fadeOut()
    })
})

```

3. 아코디언 메뉴

- index.html

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>아코디언 메뉴</title>
    <link rel="stylesheet" href="css/ex05.css">
</head>
<body>

    <nav class="gnb">
        <ul class="menu">
            <li class="d1">
                <a href="#" class="main">리바트브랜드</a>
                <div class="sub">
                    <ul>
                        <li><a href="#">리바트 이즈마인</a></li>
                        <li><a href="#">리바트 키즈</a></li>
                        <li><a href="#">H.MONDO</a></li>
                        <li><a href="#">리첸</a></li>
                        <li><a href="#">리바트 카친</a></li>
                        <li><a href="#">리바트 하음</a></li>
                        <li><a href="#">리바트 네오스</a></li>
                        <li><a href="#">리바트 벌트인</a></li>
                        <li><a href="#">리바트 마린</a></li>
                        <li><a href="#">리바트 앤솔립</a></li>
                        <li><a href="#">법인영업(H&S)</a></li>
                    </ul>
                </div>
            </li>

            <li class="d1">
                <a class="main" href="#">리바트몰</a>
                <div class="sub">
                    <ul>
                        <li><a href="#">기획전</a></li>
                        <li><a href="#">아벤트</a></li>
                        <li><a href="#">반짝딜</a></li>
                        <li><a href="#">HOT 50</a></li>
                        <li><a href="#">테마존</a></li>
                        <li><a href="#">매끼지 상품</a></li>
                        <li><a href="#">아울렛</a></li>
                    </ul>
                </div>
            </li>

            <li class="d1">
                <a class="main" href="#">리바트스토리</a>
                <div class="sub">
                    <ul>
                        <li><a href="#">리바트 미디어</a></li>
                        <li><a href="#">리바트 신제품</a></li>
                        <li><a href="#">리바트 다이어리</a></li>
                    </ul>
                </div>
            </li>
        </ul>
    </nav>

```

```

        <li><a href="#">브랜드 이벤트</a></li>
        <li><a href="#">매장 이벤트</a></li>
        <li><a href="#">주방시공사례</a></li>
    </ul>
</div>
</li>

<li class="d1">
    <a class="main" href="#">회사소개</a>
    <div class="sub">
        <ul>
            <li><a href="#">About 현대리바트</a></li>
            <li><a href="#">연구소</a></li>
            <li><a href="#">투자정보</a></li>
            <li><a href="#">인재채용</a></li>
            <li><a href="#">고객센터</a></li>
            <li><a href="#">매장찾기</a></li>
        </ul>
    </div>
</li>
</ul>
</nav>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="js/ex05.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "utf-8";

body{
    margin: 0;
}

ul{
    list-style: none;
    padding: 0;
    margin: 0;
}

a{
    text-decoration: none;
    /* 통상적으로 글꼴색으로 #000, #fff or initial은 잘 사용하지 않음, #222 등 비슷하지만 온전한 검정 또는 흰색과는 다른 색상 적용 */
    color: initial;
}

/* reset */

.gnb{
    width: 320px;
    background-color: bisque;
}

.gnb .d1{
    border-bottom: 1px solid #ccc;
}

.gnb .main{
    position: relative;
    /* 기본 inline 요소인 a태그 속성 변경 */
    display: block;
    padding: 15px;
    font-size: 18px;
    font-weight: 700;
}

/* 가상선택자로 chevron 아이콘 만들기 */
.gnb .main::after{
    position: absolute;
    right: 20px;
    top: 18px;
    content: "";
    display: block;
    width: 10px;
    height: 10px;
    border-top: 2px solid #aaa;
    border-right: 2px solid #aaa;
    transform: rotate(135deg);
}

.gnb .main.on{

```

```

        color: royalblue;
    }

/* 최상단 메뉴에 on 클래스 추가시(선택될 경우) 화살표 방향 반전 및 글자 색상 변경 */
.gnb .main.on::after{
    top: 25px;
    transform: rotate(-45deg);
}

.gnb .sub{
    display: none;
}

.gnb .sub ul{
    padding: 15px 0;
    border-top: 1px solid #ddd;
    background-color: beige;
}

.gnb .sub ul li a{
    display: block;
    padding: 5px 15px;
    font-size: 15px;
    color: #666;
}

```

- **Ex03.js**

```

/*
기본 Logic
1) 클릭해야하는 대상 : 최상단 메뉴(main)
2) 클릭한 대상의 display 값 = 'block' 일 경우 sub 메뉴 보여주기
*/

$('.gnb .main').click(function(){
    // 변수 d에 최상단 메뉴의 sub 메뉴 display 속성정보 저장
    let d = $(this).siblings('.sub').css('display')
    if(d == 'block'){
        // return을 사용하면 'null'을 리턴하게 되는데 유효성 검사에서 'null'을 받으면 통과 여부의
        // 불확실성 때문에 false를 써주는게 좋다
        return false
    }
    // 선택되는 최상단 메뉴의 sub메뉴를 펼쳐서 보여주고, 선택되지 않은 요소들은 접는다
    $('.gnb .sub').slideUp()
    $(this).siblings('.sub').slideDown()
    $('gnb .main').removeClass('on')
    $(this).addClass('on')
    return false
})

```

※ \$('셀렉터')와 HTML 구조상의 관계를 통해서 요소 접근 및 선택하기

- \$('셀렉터').parent() : 부모요소로 접근하기
- \$('셀렉터').parents() : 상위요소로 접근하기
- \$('셀렉터').siblings() : 형제요소로 접근하기
- \$('셀렉터').children() : 자식요소로 접근하기
- \$('셀렉터').find() : 자손요소로 접근하기

4. 햄버거 메뉴

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>햄버거 메뉴</title>
    <link rel="stylesheet" href="css/ex06.css">
</head>
<body>
    <div class="bttns">
        <div class="btnt1">

```

```

<p class="line"><span></span></p>
<strong>상</strong>
</div>

<div class="btnB">
<p class="line"><span></span></p>
<strong>닫</strong>
</div>

<div class="btnL">
<p class="line"><span></span></p>
<strong>좌</strong>
</div>

<div class="btnR">
<p class="line"><span></span></p>
<strong>우</strong>
</div>
</div>

<div class="sideT">
<p class="close"></p>
</div>

<div class="sideB">
<p class="close"></p>
</div>

<div class="sideL">
<p class="close"></p>
</div>

<div class="sideR">
<p class="close"></p>
</div>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="js/ex06.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "utf-8";

*{
    margin: 0;
    padding: 0;
}
/* reset */

.btns{
    display: flex;
    justify-content: space-between;
    width: 200px;
    margin: 100px auto;
}

.btns > div{
    cursor: pointer;
}

/* 가상선택자로 '닫기' 버튼 만들기 1-1(버튼영역) */
.btns .line span,
.btns .line::before,
.btns .line::after{
    display: block;
    width: 24px;
    height: 3px;
    background-color: #333;
}

.btns .line::before,
.btns .line::after{
    content: "";
}

.btns .line span{
    margin: 6px 0;
}

.sideB, .sideL, .sideR, .sideT{
    position: fixed;
    width: 100%;
    height: 100%;
}

```

```

}

.sideT{
    top: -100%;
    left: 0;
    background-color: cyan;
}

.sideB{
    bottom: -100%;
    left: 0;
    background-color: magenta;
}

.sideL{
    left: -100%;
    top: 0;
    background-color: yellow;
}

.sideR{
    right: -100%;
    top: 0;
    background-color: skyblue;
}

.close{
    position: relative;
    width: 40px;
    height: 40px;
    margin: 30px;
    background-color: #333;
    cursor: pointer;
}

/* 가상선택자로 '닫기' 버튼 만들기 1-2(버튼내부 x 아이콘) */
.close::before,
.close::after{
    position: absolute;
    top: 18px;
    left: 5px;
    content: "";
    display: block;
    width: 30px;
    height: 4px;
    background-color: #fff;
}

.close::before{
    transform: rotate(45deg);
}

.close::after{
    transform: rotate(-45deg);
}

```

- **Ex04.js**

```

// 상
$('.btnT').click(function(){
    $('.sideT').animate({
        top:0
    })
})
$('.sideT .close').click(function(){
    $('.sideT').animate({
        top: '-100%'
    })
})

// 하
$('.btnB').click(function(){
    $('.sideB').animate({
        bottom:0
    })
})
$('.sideB .close').click(function(){
    $('.sideB').animate({
        bottom: '-100%'
    })
})

```

```

// 좌
$('.btnL').click(function(){
    $('.sideL').animate({
        left:0
    })
})
$('.sideL .close').click(function(){
    $('.sideL').animate({
        left: '-100%'
    })
})

// 우
$('.btnR').click(function(){
    $('.sideR').animate({
        right:0
    })
})
$('.sideR .close').click(function(){
    $('.sideR').animate({
        right: '-100%'
    })
})

```

5. 단방향 슬라이드

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>단방향슬라이드</title>
    <link rel="stylesheet" href="css/ex07.css">
</head>
<body>
    <div id="wrap">
        <div class="photo">
            <div class="photoBox">
                
            </div>

            <div class="photoBox">
                
            </div>

            <div class="photoBox">
                
            </div>

            <div class="photoBox">
                
            </div>
        </div>
        <div class="pager">
            <ul>
                <li class="active">01</li>
                <li>02</li>
                <li>03</li>
                <li>04</li>
            </ul>
        </div>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="js/ex07.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "utf-8";

*{
    margin: 0;
    padding: 0;
}

```

```

ul{
    list-style: none;
}

#wrap{
    width: 600px;
    /* 인라인태그의 경우 display:block, 블럭요소의 경우 width값 필수 */
    margin: 50px auto;
}

.photo{
    position: relative;
    width: 600px;
    height: 300px;
    overflow: hidden;
    background-color: antiquewhite;
}

.photoBox{
    position: absolute;
    top: 0;
    left: 100%;
}

.photoBox:first-child{
    left: 0;
}

.pager{
    margin-top: 20px;
}

.pager ul{
    display: flex;
    justify-content: center;
}

.pager li {
    width: 15px;
    height: 15px;
    margin: 0 8px;
    background-color: #ccc;
    border-radius: 50%;
    /* 글자 들어쓰기 속성 */
    text-indent: -9999px;
}

.pager .active{
    background-color: lightblue;
}

```

- **Ex05.js**

```

let photoBox = $('.photoBox')
let btn = $('.pager li')
let c = 0 // 화면에 출력중인 이미지의 순서값을 저장(기준점을 잡는다)

btn.click(function(){
    let i = $(this).index()
    // 조건문이 한줄인데 return을 사용할 경우 중괄호 생략가능
    if(i == c) return false
    btn.removeClass('active')
    $(this).addClass('active')
    photoBox.eq(c).stop().animate({
        left: '-100%'
    })
    photoBox.eq(i).css({
        left: '100%'
    }).stop().animate({
        left : 0
    })
    c = i
})

```

6. 양방향 슬라이드 (좌우, 상하)

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>양방향 슬라이드</title>
    <link rel="stylesheet" href="../3일차/css/ex08.css">
</head>
<body>
    <div id="wrap">
        <div class="photo">
            <div class="photoBox">
                
            </div>

            <div class="photoBox">
                
            </div>

            <div class="photoBox">
                
            </div>

            <div class="photoBox">
                
            </div>
        </div>
        <div class="pager">
            <ul>
                <li class="active">01</li>
                <li>02</li>
                <li>03</li>
                <li>04</li>
            </ul>
        </div>
    </div>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="../3일차/js/ex08.js"></script>
</body>
</html>

```

• style.css

```

@charset "utf-8";

*{
    margin: 0;
    padding: 0;
}

ul{
    list-style: none;
}

#wrap{
    width: 600px;
    /* 인라인태그의 경우 display:block, 블럭요소의 경우 width값 필수 */
    margin: 50px auto;
}

.photo{
    position: relative;
    width: 600px;
    height: 300px;
    overflow: hidden;
    background-color: antiquewhite;
}

.photoBox{
    position: absolute; /* 상하 */
    left: -100%;          /* top: -100% */
    top: 0;               /* left: 0 */
}

.photoBox:first-child{
    left: 0;              /* top: 0% */
}

.pager{
    margin-top: 20px;
}

```

```

.pager ul{
    display: flex;
    justify-content: center;
}

.pager li {
    width: 15px;
    height: 15px;
    margin: 0 8px;
    background-color: #ccc;
    border-radius: 50%;
    /* 글자 들여쓰기 속성 */
    text-indent: -9999px;
}

.pager .active{
    background-color: lightblue;
}

```

- **Ex06.js**

```

let image = $('.photoBox')
let pager = $('.pager li')
let c = 0

// ('셀렉터').이벤트() : 스크립트에 의해 추가되는 영역(동적인 영역)에는 이벤트 적용 안됨
// ('셀렉터').on(이벤트) : 스크립트에 의해 추가되는 영역(동적인 영역)에도 이벤트 적용 가능
pager.on({
    click: function () {
        let i = $(this).index();
        // 현재 화면상의 이미지
        let currentImg = image.eq(c);
        // 클릭으로 화면에 들어올 예정인 이미지
        let newImg = image.eq(i);

        // 변수 i와 c의 값을 비교해서 동작 방향을 다르게 설정
        // i == c -> 활성화 중인 버튼을 클릭했다. 여기서 명령 종료
        // i > c -> 활성화 중인 버튼의 오른쪽 버튼을 클릭. 위에서 아래으로 이동
        // i < c -> 활성화 중인 버튼의 왼쪽 버튼을 클릭. 아래에서 위로 이동
        // 좌우
        if (i == c) return false;
        if (i > c) {
            currentImg.stop().animate({ left: '-100%' });
            newImg.css({ left: '100%' }).stop().animate({ left: 0 });
        }
        if (i < c) {
            currentImg.stop().animate({ left: '100%' });
            newImg.css({ left: '-100%' }).stop().animate({ left: 0 });
        }

        // 상하
        /*      if (i == c) return false;
        if (i > c) {
            currentImg.stop().animate({ top: '100%' });
            newImg.css({ top: '-100%' }).stop().animate({ top: 0 });
        }
        if (i < c) {
            currentImg.stop().animate({ top: '-100%' });
            newImg.css({ top: '100%' }).stop().animate({ top: 0 });
        }*/
        c = i;
        pager.removeClass('active');
        pager.eq(i).addClass('active');
    },
});

```

7. 슬라이드 좌우버튼

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>슬라이드 좌우버튼</title>
    <link rel="stylesheet" href="css/ex09.css">
</head>

```

```

<body>
    <div class="slide">
        <div class="sImg">
            <div class="sImgBox">
                <a href="">
                    
                </a>
            </div>
            <div class="sImgBox">
                <a href="">
                    
                </a>
            </div>
            <div class="sImgBox">
                <a href="">
                    
                </a>
            </div>
            <div class="sImgBox">
                <a href="">
                    
                </a>
            </div>
        </div>
        <div class="sPager">
            <ul>
                <li>01</li>
                <li>02</li>
                <li>03</li>
                <li>04</li>
            </ul>
        </div>
        <div class="sPrev">
            <span>이전으로</span>
        </div>
        <div class="sNext">
            <span>다음으로</span>
        </div>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="js/ex09.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "UTF-8";

*{
    margin: 0;
    padding: 0;
}

ul{
    list-style: none;
}

.slide{
    position: relative;
    min-width: 1240px;
}

.sImg{
    position: relative;
    height: 420px;
    overflow: hidden;
}

.sImgBox{
    position: absolute;
    top: 0;
    left: 100%;
    width: 100%;
    height: 420px;
    overflow: hidden;
}

.sImgBox a{
    position: absolute;
    top: 0;
    left: 50%;
    margin-left: -960px;
}

```

```

.sPager{
    margin-top: 30px;
}

.sPager ul{
    display: flex;
    justify-content: center;
}

.sPager li{
    width: 50px;
    height: 5px;
    margin: 0 10px;
    text-indent: -99999px;
    cursor: pointer;
    background-color: #ccc;
}

.sPager .active{
    background-color: skyblue;
}

.sPrev, .sNext{
    position: absolute;
    top: 180px;
    width: 60px;
    height: 60px;
    border-top: 4px solid #555;
    border-left: 4px solid #555;
    cursor: pointer;
}

.sPrev{
    left: 100px;
    transform: rotate(-45deg);
}

.sNext{
    right: 100px;
    transform: rotate(135deg);
}

.sPrev span, .sNext span{
    display: block;
    width: 0;
    height: 0;
    opacity: 0;
    overflow: hidden;
}

```

- **Ex07.js**

```

let s = 2;
let photo = $('.sImgBox')
let pager = $('.sPager ul li')
let btnPrev = $('.sPrev')
let btnNext = $('.sNext')
let all = photo.size()
// size() : 선택된 자료의 개수를 저장한다.

// 초기값 세팅
photo.eq(s).css({
    left: 0
})
pager.eq(s).addClass('active')

// 페이지 버튼 클릭
pager.on({
    click: function(){
        let i = $(this).index()
        pager.removeClass('active')
        $(this).addClass('active')
        if (i > s) moveToLeft(i)
        if (i < s) moveToRight(i)
    }
})

// 좌우 버튼
btnPrev.on({
    click : function(){
        let n = s - 1
        if (n < 0) n = all - 1
        pager.eq(n).click()
    }
})

```

```

        })
    btnNext.on({
        click: function(){
            let n = s + 1
            if(n == all) n = 0
            pager.eq(n).click()
        }
    })

    // 함수 정의
    function moveToLeft(j){
        let currentPhoto = photo.eq(s)
        let newPhoto = photo.eq(j)
        currentPhoto.stop().animate({left: '-100%'}, 1500)
        newPhoto.css({left: '100%'}).stop().animate({left: 0}, 1500)
        s = j
    }

    function moveToRight(j){
        let currentPhoto = photo.eq(s)
        let newPhoto = photo.eq(j)
        currentPhoto.stop().animate({left: '100%'}, 1500)
        newPhoto.css({left: '-100%'}).stop().animate({left: 0}, 1500)
        s = j
    }
}

```

- `size()` : 선택된 자료의 개수를 저장

8. 슬라이드 자동재생

- `index.html`

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>슬라이드 자동재생</title>
    <link rel="stylesheet" href="css/ex10.css">
</head>
<body>
    <div class="slide">
        <div class="sImg">
            <div class="sImgBox">
                <a href="">
                    
                </a>
            </div>
            <div class="sImgBox">
                <a href="">
                    
                </a>
            </div>
            <div class="sImgBox">
                <a href="">
                    
                </a>
            </div>
            <div class="sImgBox">
                <a href="">
                    
                </a>
            </div>
        </div>
        <div class="sPager">
            <ul>
                <li>01</li>
                <li>02</li>
                <li>03</li>
                <li>04</li>
            </ul>
            <p class="sPlay"><span>재생</span></p>
            <p class="sStop"><span>중지</span></p>
        </div>
        <div class="sPrev">
            <span>이전으로</span>
        </div>
        <div class="sNext">
            <span>다음으로</span>
        </div>
    </div>
    <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>

```

```
<script src="js/ex10.js"></script>
</body>
</html>
```

- **style.css**

```
@charset "UTF-8";

*{
    margin: 0;
    padding: 0;
}

ul {
    list-style: none;
}

.slide{
    position: relative;
    min-width: 1240px;
}

.sImg{
    position: relative;
    height: 420px;
    overflow: hidden;
}

.sImgBox{
    position: absolute;
    top: 0;
    left: 100%;
    width: 100%;
    height: 420px;
    overflow: hidden;
}

.sImgBox a{
    position: absolute;
    top: 0;
    left: 50%;
    margin-left: -960px;
}

.sPager{
    display: flex;
    justify-content: center;
    margin-top: 30px;
}

.sPager ul{
    display: flex;
}

.sPager li{
    width: 50px;
    height: 5px;
    margin: 0 10px;
    text-indent: -99999px;
    cursor: pointer;
    background-color: #ccc;
}

.sPager .active{
    background-color: skyblue
}

.sPrev, .sNext{
    position: absolute;
    top: 180px;
    width: 60px;
    height: 60px;
    border-top: 4px solid #555;
    border-left: 4px solid #555;
    cursor: pointer;
}

.sPrev{
    left: 100px;
    transform: rotate(-45deg);
}

.sNext{
    right: 100px;
}
```

```

        transform: rotate(135deg);
    }

.sPrev span, .sNext span, .sPlay span, .sStop span{
    display: block;
    width: 0;
    height: 0;
    opacity: 0;
    overflow: hidden;
}

.sPlay, .sStop{
    width: 25px;
    height: 25px;
    margin-left: 20px;
    cursor: pointer;
}

.sPlay{
    display: none;
    background-image: url(../../../../23일차/Ex/img	btn_play.png);
}

.sStop{
    background-image: url(../../../../23일차/Ex/img	btn_pause.png);
}

```

- **Ex08.js**

```

let s = 2;
let photo = $('.sImgBox')
let pager = $('.sPager ul li')
let btnPrev = $('.sPrev')
let btnNext = $('.sNext')
let all = photo.size()
let play = $('.sPlay')
let stop = $('.sStop')
let autoPlay

// 초기값 세팅
photo.eq(s).css({
    left: 0
})
pager.eq(s).addClass('active')

// 페이지 클릭
pager.on({
    click: function(){
        let j = $(this).index()
        pager.removeClass('active')
        $(this).addClass('active')
        if(j > s) moveToLeft(j)
        if(j < s) moveToRight(j)
    }
})

// 좌우 버튼
btnPrev.on({
    click: function(){
        let n = s - 1
        if (n < 0) n = all - 1
        pager.eq(n).click()
    }
})
btnNext.on({
    click: function(){
        let n = s + 1
        if (n == all) n = 0
        pager.eq(n).click()
    }
})

// 재생/중지 버튼
stop.on({
    click: function(){
        clearInterval(autoPlay)
        stop.hide()
        play.show()
    }
})
play.on({
    click: function(){
        timer()
        play.hide()
    }
})

```

```

        stop.show()
    }
})

// 함수 생성
function moveToLeft(i){
    let currentPhoto = photo.eq(s)
    let newPhoto = photo.eq(i)
    currentPhoto.stop().animate({left: '-100%'}, 1000)
    newPhoto.css({left: '100%'}).stop().animate({left: 0},1000)
    s = i
}
function moveToRight(i){
    let currentPhoto = photo.eq(s)
    let newPhoto = photo.eq(i)
    currentPhoto.stop().animate({left:'100%'},1000)
    newPhoto.css({left: '-100%'}).stop().animate({left: 0},1000)
    s = i
}

timer()
function timer(){
    autoPlay = setInterval(function(){
        let n = s + 1
        if (n == all) n = 0
        pager.eq(n).click()
    },3000)
}

```

- 자동재생 명령

⇒ **setInterval(함수, 시간)**

⇒ 설정한 시간의 간격마다 함수 명령을 반복해서 실행한다

- 자동재생 멈춤

⇒ **clearInterval(변수)**

⇒ setInterval() 단독으로 사용해도 실행되긴 하지만, 만약에 clearInterval()을 사용해서 자동재생을 컨트롤 해줘야하는 경우라면 clearInterval()을 저장해둘 필요가 없다

```

let 변수 = setInterval(함수, 시간)
clearInterval()

```

9. Slick 플러그인 #1

- jQuery, slick.css, slick-theme.css, slick.min.js 연결 : <https://www.codingfactory.net/12845> , <https://kenwheeler.github.io/slick/>

• 슬릭-슬라이더-사용법-및-옵션 : <https://velog.io/@rgfdds98/jQuery-slick-슬라이더-사용법-및-옵션>

⇒ slick.css와 slick.js 다운로드하여 연결

⇒ **주의사항**: 각 슬라이드에 고유 클래스명(slide1, slide2, slide3 등 사용자 설정) 사용할 것

- `:nth-child` 는 예리남

⇒ 선택자 연결은 slide를 감싸고 있는 slide-group(상위 태그)과 연결할 것

```

$('.slider_area').slick({
    rows: 1,           //이미지를 몇 줄로 나타낼것지 (int)
    dots: false,        //네비게이션버튼 (boolean) -default:false
    appendDots: $('#dots'), //네비게이션버튼 변경 (선택자 혹은 $element)
    dotsClass: 'custom-dots', //네비게이션버튼 클래스 변경
    infinite: true,    //무한반복 (boolean) -default:true
    slidesToShow: 4,     //한번에 보여줄 사진의 갯수(int)
    slidesToScroll: 4,   //한번에 넘길 사진의 갯수(int)
    slidesPerRow: 1,     //보여질 행의 수 (한 줄, 두 줄 ... )
    autoplay: true,      //자동시작 (boolean) -default:false
    autoplaySpeed: 2000, //자동날마다 시간(int, 1000ms = 1초)
    variableWidth: true, //사진마다 width의 크기가 다른가? (boolean) -default:false
    draggable: false,   //리스트 드래그 가능여부 (boolean) -default:true
    arrows: true,        //화살표(날기기버튼) 여부 (boolean) -default:true
    pauseOnFocus: true,  //마우스 포커스 시 슬라이드 멈춤 -default:true
    pauseOnHover: true,  //마우스 호버 시 슬라이드 멈춤 -default:true
    pauseOnDotsHover: true, //네이게이션버튼 호버 시 슬라이드 멈춤 -default:false
    vertical: true,      //세로방향 여부 (boolean) -default:false
    verticalSwiping: true, //세로방향 스와이프 여부 (boolean) -default:false
    accessibility: true, //접근성 여부 (boolean) -default:true
    appendArrows: $('#arrows'), //좌우 화살표 변경 (선택자 혹은 $element)
    prevArrow: $('#prevArrow'), //좌(이전) 화살표만 변경 (선택자 혹은 $element)
    nextArrow: $('#nextArrow'), //우(다음) 화살표만 변경 (선택자 혹은 $element)
    initialSlide: 1,        //처음 보여질 슬라이드 번호 -default:0
    centerMode: true,       //중앙에 슬라이드가 보여지는 모드 -default:false
    centerPadding: '70px', //중앙에 슬라이드가 보여지는 모드에서 padding값
    fade: true,             //모션 사용 여부 -default:false
    speed: 2000,            //모션 시간 (얼마나 빨름속도로 넘어가는지)(int, 1000ms = 1초)
    waitForAnimate: true,   //애니메이션 중에는 동작을 재활용 -default:true
    responsive: [
        {
            breakpoint: 1024, //breakpoint: 반응형 구간
            settings: { //settings: 반응형 구간에 따른 설정 변경
                slidesToShow: 3,
                slidesToScroll: 3
            }
        },
        {
            breakpoint: 480,
            settings: {
                slidesToShow: 2,
                slidesToScroll: 2
            }
        }
    ] //반응형
});

```

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>slick 플러그인 01</title>
    <link rel="stylesheet" href="css/slick.css">
    <link rel="stylesheet" href="css/ex11.css">
</head>
<body>
    <h1>slick 기본 형태 (자동x, 좌우버튼o)</h1>
    <div class="slide s01">
        <div></div>
        <div></div>
        <div></div>
        <div></div>
    </div>
    <br>
    <h1>좌우버튼o, 페이저o</h1>
    <div class="slide s02">
        <div></div>
        <div></div>
        <div></div>
        <div></div>
    </div>
    <br>
    <h1>좌우버튼o, 페이저o, 자동재생</h1>
    <div class="slide s03">
        <div></div>
        <div></div>
        <div></div>

```

```

<div></div>
</div>
<br>
<h1>케라셀</h1>
<div class="slide s04">
    <div class="s04Box"></div>
    <div class="s04Box"></div>
    <div class="s04Box"></div>
    <div class="s04Box"></div>
    <div class="s04Box"></div>
</div>
<br>
<h1>커스터마이징 버튼</h1>
<div class="slide">
    <div class="slideImg s05">
        <div class="slideImgBox">
            <a href=""></a>
        </div>
        <div class="slideImgBox">
            <a href=""></a>
        </div>
        <div class="slideImgBox">
            <a href=""></a>
        </div>
        <div class="slideImgBox">
            <a href=""></a>
        </div>
    </div>
    <div class="slideBtn">
        <div class="pager"></div>
        <button class="btnPrev">이전으로</button>
        <button class="btnNext">다음으로</button>
    </div>
</div>
<script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
<script src="js/slick.min.js"></script>
<script src="js/ex11.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "utf-8";

*{
    margin: 0;
    padding: 0;
}

ul{
    list-style: none;
}

img{
    /* img 태그는 기본적으로 block 요소로 감싸고 해당 block의 width, height 옵션 적용
    img 태그는 절대 직접적으로 효과를 주지 말것 */
    vertical-align: top;
    width: 100%;
    height: auto;
}

h1{
    width: 600px;
    margin: 20px auto;
    font-size: 20px;
    /* 폰트사이즈가 20px이 넘어가면 line-height를 최소 1.2em이상 사용할것 */
    line-height: 1.2em;
}

.slide{
    position: relative;
    width: 600px;
    margin: 0 auto 50px;
}

.s04Box{
    box-sizing: border-box;
    padding: 0 5px;
}

```

- **Ex09-1.js**

```

// slick 기본 형태 (자동x, 좌우버튼o)
$('.s01').slick()

// 좌우버튼o, 페이저o
$('.s02').slick({
    // pager 버튼 생성
    dots: true
})

// 좌우버튼o, 페이저o, 자동재생
$('.s03').slick({
    dots: true,
    // 자동재생여부 결정
    autoplay: true,
    // 자동재생 되어지는 간격에 대한 속성
    autoplaySpeed: 3000,
    // 이동속도
    speed: 1000
})

// 케러셀
$('.s04').slick({
    // 한번에 보여지는 카드의 개수
    slidesToShow: 3,
    // 한번에 이동하는 카드의 수
    slidesToScroll: 2,
})

// 커스터마이징 버튼
$('.s05').slick({
    prevArrow: '.slideBtn .btnPrev',
    nextArrow: '.slideBtn .btnNext',
    dots: true,
    appendDots: '.slideBtn .pager'
})

```

- arrows 옵션 : 좌우버튼, 기본값 true
- prevArrow 옵션 : 이전버튼으로 사용할 대상을 선택
- nextArrow 옵션 : 다음버튼으로 사용할 대상을 선택
- appendDots 옵션 : 선택한 태그 안쪽에 페이지 생성

9. Slick 플러그인 #2

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>slick 플러그인 #2</title>
    <link rel="stylesheet" href="css/slick.css">
    <link rel="stylesheet" href="css/ex12.css">
</head>
<body>
    <div class="mainSlide">
        <div class="msImg">
            <div class="msImgBox">
                <a href="">
                    <p class="photo"></p>
                </a>
            </div>
            <div class="msImgBox">
                <a href="">
                    <p class="photo"></p>
                </a>
            </div>
            <div class="msImgBox">
                <a href="">
                    <p class="photo"></p>
                </a>
            </div>
            <div class="msImgBox">
                <a href="">
                    <p class="photo"></p>
                </a>
            </div>
        </div>
    </div>

```

```

        </div>
    </div>
    <p class="msPrev">이전으로</p>
    <p class="msNext">다음으로</p>
    <div class="msBtn">
        <div class="msPager"></div>
        <p class="msPause">멈춤</p>
        <p class="msPlay">재생</p>
    </div>
    <!-- 슬릭 플러그인에서 사용 할 버튼영역에는 a태그 사용 불가 -->
</div>
<script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
<script src="js/slick.min.js"></script>
<script src="js/ex12.js"></script>
</body>
</html>

```

- **style.css**

```

@charset "utf-8";

*{
    margin: 0;
    padding: 0;
}

ul{
    list-style: none;
}

img{
    vertical-align: top;
}

h1{
    font-size: 22px;
    line-height: 1em;
    margin: 20px 0;
}

.mainSlide{
    position: relative;
    min-width: 1240px;
    margin-bottom: 100px;
}

.msImgBox{
    overflow: hidden;
}

.msImgBox a{
    display: block;
    width: 1200px;
    margin: 0 auto;
}

.msImgBox a .photo{
    margin-left: -360px;
}

.mainSlide .msPrev,
.mainSlide .msNext{
    position: absolute;
    top: 170px;
    width: 80px;
    height: 80px;
    border: 1px solid #777;
    border-radius: 50%;
    text-indent: -9999999px;
    cursor: pointer;
    background-repeat: no-repeat;
    background-position: center;
    background-size: 22px auto;
    background-color: #eddede;
}

.mainSlide .msPrev{
    left: 50%;
    margin-left: -640px;
    background-image: url(../../../../23일차/Ex/img/btn_prev.png);
}

.mainSlide .msNext{
    right: 50%;
}

```

```

        margin-right: -640px;
        background-image: url(../../../../23일차/Ex/img	btn_next.png);
    }

.msBtn{
    position: absolute;
    bottom: 20px;
    left: 0;
    display: flex;
    justify-content: center;
    width: 100%;
}

.msPager ul{
    display: flex !important;
    /* 우선순위를 무시하고 !important 키워드를 붙인값을 최종결과로 브라우저에 반영해줌 */
}

.msPager li{
    padding: 5px;
}

.msPager li button{
    vertical-align: top;
    width: 16px;
    height: 16px;
    border: 0;
    border-radius: 50px;
    text-indent: -999999px;
    background-color: #aaa;
    cursor: pointer;
}

.msPager .slick-active button{
    background-color: red;
}

.msPause, .msPlay{
    width: 24px;
    height: 24px;
    margin-left: 10px;
    text-indent: -999999px;
    cursor: pointer;
}

.msPause{
    background-image: url(../../../../23일차/Ex/img	btn_pause.png);
}

.msPlay{
    background-image: url(../../../../23일차/Ex/img	btn_play.png);
    display: none;
}

```

• Ex09-2.js

```

$('.msImg').slick({
    prevArrow: '.mainSlide .msPrev',
    nextArrow: '.mainSlide .msNext',
    dots: true,
    appendDots: '.mainSlide .msPager',
    autoplay: true,
    autoplaySpeed: 3000,
    speed: 1000,
    fade: true
})

$('.mainSlide .msPause').click(function(){
    $('.msImg').slick('slickPause')
    $(this).hide()
    $('.mainSlide .msPlay').show()
})

$('.mainSlide .msPlay').click(function(){
    $('.msImg').slick('slickPlay')
    $(this).hide()
    $('.mainSlide .msPause').show()
})

```

9. Slick 플러그인 #3 - 모바일 전용

- slick과 함께 사용할 영역에는 a태그 사용 안함
- Vertical-align : 인라인 블록 등을 포함한 모든 인라인요소의 수직 정렬을 위해 사용

vertical-align	
속성값	baseline sub super top text-top middle bottom text-bottom <percentage> <length> inherit
기본값	baseline
적용대상	인라인 요소와 테이블 셀
상속여부	No
퍼센트	요소의 line-height 값 참조
산출값	길이와 퍼센트 같은 절대 길이, 이외는 지정한 값

- slick 사용시 아래와 같이 자체 클래스 이름 부여

```

    ▶ <div class="slick-list draggable"> ⓘ </div>
      ▼ <ul class="slick-dots" style="display: block;" role="tablist"> == $0
        ▶ <li class="slick-active" role="presentation"> ⓘ </li>
  
```

- !important : 말 뜻 그대로 '중요한 속성' 을 의미하고, 해당 속성이 변경되지 않도록 하는 역할을 한다.

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title></title>
6   <style>
7     p{
8       background-color: red !important;
9       float: left;
10    }
11
12    p{
13      background-color: blue;
14      float: left
15    }
16  </style>
17 </head>
18 <body>
19   <p>important</p>
20 </body>
21 </html>
  
```

7번 라인과 12번 라인에 <p> 태그의 css를 정의하였다.

!important 속성이 없다면 <p> 태그의 내용은 12번

라인에 의해 파란색이 되어야 하지만

!important 속성에 의해 수정이 되지 않는 것을 확인할 수 있다.

- slick('setPosition') : display:none 처리가 된 슬라이드 영역을 화면에 노출시키면, 슬릭영역이 제대로 자리를 잡지 못한다.

최초 한번 슬라이드가 동작한 후에는 슬라이드가 정상적으로 자리를 잡는다.

그래서 setPosition 옵션을 사용하도록 해두었다

- index.html

```

<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  
```

```

<title>slick 플러그인 03 - 모바일 전용</title>
<link rel="stylesheet" href="../5일차/css/slick.css">
<link rel="stylesheet" href="css/ex13.css">
</head>
<body>
    <div id="wrap">
        <div class="product">
            <div class="pdTabBtn">
                <ul>
                    <li class="active">음료</li>
                    <li>화장품</li>
                    <li>카카오</li>
                    <!-- slick과 함께 사용할 영역에는 a태그 사용 안함 -->
                </ul>
            </div>
            <div class="pdTabPannel">
                <!-- 음료 -->
                <div class="pdList view">
                    <div class="pdItem">
                        <a href=""></a>
                    </div>
                    <div class="pdItem">
                        <a href=""></a>
                    </div>
                </div>
                <!-- 화장품 -->
                <div class="pdList">
                    <div class="pdItem">
                        <a href=""></a>
                    </div>
                    <div class="pdItem">
                        <a href=""></a>
                    </div>
                </div>
                <!-- 카카오 -->
                <div class="pdList">
                    <div class="pdItem">
                        <a href=""></a>
                    </div>
                    <div class="pdItem">
                        <a href=""></a>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
    <script src="../5일차/js/slick.min.js"></script>
    <script src="js/ex13.js"></script>
</body>
</html>

```

• style.css

```

@charset "utf-8";
*{

```

```

        margin: 0;
        padding: 0;
    }

    ul{
        list-style: none;
    }

    img{
        vertical-align: top;
    }

    .pdTabBtn{
        margin-top: 50px;
        margin-bottom: 30px;
    }

    .pdTabBtn ul{
        display: flex;
        justify-content: center;
    }

    .pdTabBtn li{
        margin: 5px;
        padding: 5px 15px;
        border: 1px solid #aaa;
        border-radius: 50px;
        font-size: 14px;
        color: #555;
        cursor: pointer;
    }

    .pdTabBtn .active{
        border-color: green;
        background-color: green;
        color: #fff;
    }

    .pdTabPannel{
        overflow: hidden;
    }

    .pdList{
        display: none;
        width: 160%;
        padding: 0 20px;
    }

    .pdList.view{
        display: block;
    }

    .pdItem{
        box-sizing: border-box;
        padding: 0 6px;
    }

    .pdItem img{
        width: 100%;
        height: auto;
    }

    .pdList ul.slick-dots{
        display: flex !important;
        justify-content: center;
        width: 60%;
        margin-top: 20px;
    }

    .pdList ul.slick-dots li{
        margin: 0 5px;
    }

    .pdList ul.slick-dots li button{
        width: 12px;
        height: 12px;
        border: 0;
        border-radius: 50%;
        background-color: #aaa;
        text-indent: -9999999px;
    }

    .pdList ul.slick-dots .slick-active button{
        background-color: skyblue;
    }

```

- Ex09-3.js

```
$('.pdTabBtn ul li').click(function(){
    let i = $(this).index()
    $('.pdTabBtn ul li').removeClass('active')
    $(this).addClass('active')
    $('.pdList').removeClass('view')
    $('.pdList').eq(i).addClass('view')
    // display:none 처리가 된 슬라이드 영역을 화면에 노출시키면, 슬릭영역이 제대로 자리를 잡지 못한다.
    // 최초 한번 슬라이드가 동작한 후에는 슬라이드가 정상적으로 자리를 잡는다.
    // 그래서 setPosition 옵션을 사용하도록 해두었다
    $('.pdList').slick('setPosition')
})

$('.pdList').slick({
    arrows: false,
    dots: true,
    slidesToShow: 2
})
```

9. Slick 플러그인 #4 - 반응형

- Ex09-4.js

```
$('.pdList').slick({
    arrows: false,
    dots: true,
    slidesToShow: 2,
    // 반응형 작업시에 모바일이 먼저 계산되도록 설정!
    mobileFirst: true,
    responsive: [
        {
            // 화면너비값이 767px를 초과하였을 때 설정
            breakpoint: 767,
            settings: {
                slidesToShow: 4
            }
        },
        {
            breakpoint: 1200,
            settings: {
                slidesToShow: 5,
                dots: false
            }
        }
    ]
})
```

10. swiper 플러그인

- 참조 : <https://swiperjs.com/> , <https://shadesign.tistory.com/entry/swiper-slide-총정리사용법-적용-옵션>
- 업데이트 빈도가 잦은편이라 파일로 다운로드 권장
- 사용방법

```
let 변수이름 = new Swiper('.swiper-wrapper의 부모태그',{
    옵션이름 : '값',
    옵션이름 : '값',
    옵션이름 : {
        옵션2이름 : '값',
        옵션2이름 : '값',
    }
})
```

- index.html

```
<!DOCTYPE html>
<html lang="ko">
```

```

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>swiper 퀄리고</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/swiper@9/swiper-bundle.min.css"/>
    <link rel="stylesheet" href="css/ex15.css">
</head>
<body>
    <div class="slide_wrap">
        <div class="slide_list">
            <!-- 슬라이드 이미지 전체를 감싸고 있는 틀은 반드시 swiper-wrapper를 사용해야 한다 -->
            <div class="swiper-wrapper">
                <!-- 마찬가지로 각 슬라이드 영역을 감싸는 틀도 반드시 swiper-slide를 사용해야 함 -->
                <div class="swiper-slide"><a href=""></a></div>
                <div class="swiper-slide"><a href=""></a></div>
                <div class="swiper-slide"><a href=""></a></div>
                <div class="swiper-slide"><a href=""></a></div>
            </div>
        </div>
        <div class="pager"></div>
        <div class="btn_L"></div>
        <div class="btn_R"></div>
        <div class="scrollbar"></div>
    </div>
    <br><hr>
    <div class="product_wrap">
        <div class="product_list">
            <div class="swiper-wrapper">
                <div class="swiper-slide"><a href=""></a></div>
                <div class="swiper-slide"><a href=""></a></div>
                <div class="swiper-slide"><a href=""></a></div>
                <div class="swiper-slide"><a href=""></a></div>
                <div class="swiper-slide"><a href=""></a></div>
            </div>
        </div>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/swiper@9/swiper-bundle.min.js"></script>
    <script src="js/ex15.js"></script>
</body>
</html>

```

• style.css

```

@charset "utf-8";

*{
    margin: 0;
    padding: 0;
}

ul{
    list-style: none;
}

img{
    vertical-align: top;
}

.slide_wrap{
    position: relative;
    width: 600px;
    margin: 50px auto;
}

.slide_list{
    overflow: hidden;
}

.slide_wrap .pager{
    display: flex;
    justify-content: center;
    margin-top: 30px;
}

.slide_wrap .pager span{
    display: block;
    width: 15px;
    height: 15px;
    border-radius: 50%;
    background-color: #777;
}

```

```

.slide_wrap .pager .swiper-pagination-bullet-active {
    background-color: blueviolet;
}

.slide_wrap .btn_L, .slide_wrap .btn_R{
    position: absolute;
    top: 115px;
    width: 70px;
    height: 70px;
    border: 1px solid #aaa;
    border-radius: 50%;
    background-repeat: no-repeat;
    background-position: center;
    background-size: 14px auto;
    cursor: pointer;
}

.slide_wrap .btn_L{
    left: -100px;
    background-image: url(../../../../23일차/Ex/img	btn_prev.png);
}

.slide_wrap .btn_R{
    right: -100px;
    background-image: url(../../../../23일차/Ex/img	btn_next.png);
}

.slide_wrap .scrollbar{
    width: 600px;
    height: 7px;
    margin: 20px auto;
    background-color: #ddd;
}

.slide_wrap .scrollbar div{
    background-color: blueviolet;
}

.product_wrap{
    width: 80%;
    margin: 50px auto;
}

.product_list{
    overflow: hidden;
}

.product_wrap img{
    width: 100%;
    height: auto;
}

```

- **Ex10-1.js**

```

const slide01 = new Swiper('.slide_list', {
    loop: true,
    pagination: {
        el: '.slide_wrap .pager',
        clickable : true
    },
    navigation: {
        nextEl : '.slide_wrap .btnR',
        prevEl : '.slide_wrap .btnL'
    },
    // scroll bar는 loop:true와 함께 사용할 수 없다
    scrollbar: {
        el: '.slide_wrap .scrollbar'
    },
    // 자동재생
    autoplay: {
        delay: 4000
    },
    // 슬라이드 움직이는 속도
    speed: 1000
})

const pdList = new Swiper('.product_list', {
    // 슬라이드 보여지는 개수
    slidesPerView : 2,
    // 슬라이드 양 옆에 마진(여백) 주기
    spaceBetween : 3
})

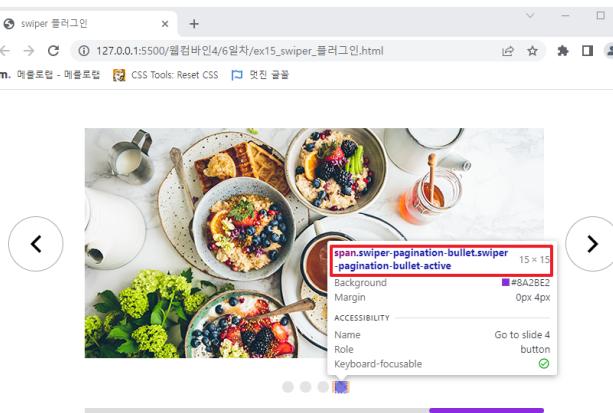
```

※ swiper 임의의 클래스 이름 부여

```

.ex15.css - 파일 없음(작업 영역) - Visual Studio Code
ex14.js ex15_swiper_플러그인.html ex15.css ex15.js ...
① BSW > 웹컴바인4 > 6일자 > css > ex15.css ...
30 }
31 .slide_wrap .pager span{
32   display: block;
33   width: 15px;
34   height: 15px;
35   border-radius: 50%;
36   background-color: #777;
37 }
38 }
39 .slide_wrap .pager .swiper-pagination-bullet-active {
40   background-color: blueviolet;
41 }
42 }
43 .slide_wrap .btn_L, .slide_wrap .btn_R{
44   position: absolute;
45   top: 115px;
46   width: 70px;
47   height: 70px;
48   border: 1px solid #aaa;
49   border-radius: 50%;
50   background-repeat: no-repeat;
51   background-position: center;
52   background-size: 14px auto;
53   cursor: pointer;
54 }
55 }

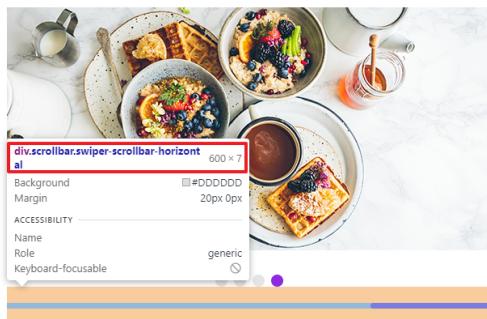
```



```

<body>
  <div class="slide_wrap">
    <div class="slide_list">
      <!-- 슬라이드 이미지 전체를 감싸고 있는 블록은 반드시 swiper-wrapper를 사용해야 한다 -->
      <div class="swiper-wrapper">
        <!-- 미간가지로 각 슬라이드 영역을 감싸는 블록도 반드시 swiper-slide를 사용해야 함 -->
        <div class="swiper-slide"><a href=""></a></div>
        <div class="swiper-slide"><a href=""></a></div>
        <div class="swiper-slide"><a href=""></a></div>
        <div class="swiper-slide"><a href=""></a></div>
      </div>
    </div>
  </div>

```



```

... ▼ <div class="slide_list swiper-initialized swiper-horizontal swiper-backface-hidden" style="position: relative; width: 100%; height: 100%;"> == $0
  <!-- 슬라이드 이미지 전체를 감싸고 있는 블록은 반드시 swiper-wrapper를 사용해야 한다 -->
  > <div class="swiper-wrapper" id="swiper-wrapper-9eb2840b21009e95d" aria-live="off" style="transition-duration: 1000ms; transform: translate3d(-1800px, 0px, 0px);> ... </div> flex
    <span class="swiper-notification" aria-live="assertive" aria-atomic="true"></span>
  </div>
  ▼ <div class="pager swiper-pagination-clickable swiper-pagination-horizontal" style="text-align: center; margin-top: 10px; font-size: 0.8em; font-weight: bold; font-style: italic; font-family: sans-serif; position: relative; width: 100%; height: 100%;>
    <span class="swiper-pagination-bullet" tabindex="0" role="button" aria-label="Go to slide 1"></span>
    <span class="swiper-pagination-bullet" tabindex="0" role="button" aria-label="Go to slide 2"></span>
    <span class="swiper-pagination-bullet swiper-pagination-bullet-active" tabindex="0" role="button" aria-label="Go to slide 3" aria-current="true"></span>
    <span class="swiper-pagination-bullet" tabindex="0" role="button" aria-label="Go to slide 4" aria-current="false"></span>
  </div>

```

10. swiper 플페이지

- index.html

```

<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>16 swiper full page</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/swiper@9/swiper-bundle.min.css"/>
  <link rel="stylesheet" href="css/ex16.css">
</head>
<body>
  <div id="wrap">
    <header id="header">헤더</header>
    <main id="container">
      <div class="contList">
        <div class="swiper-wrapper">
          <section class="swiper-slide">
            <div class="main m1">섹션1</div>
          </section>

```

```

        <section class="swiper-slide">
            <div class="main m2">섹션2</div>
        </section>
        <section class="swiper-slide">
            <div class="main m3">섹션3</div>
        </section>
    </div>
    </div>
    <div class="contPager"></div>
</main>
<footer id="footer">푸터</footer>
</div>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/swiper@9/swiper-bundle.min.js"></script>
<script src="js/ex16.js"></script>
</body>
</html>

```

• **style.css**

```

@charset "utf-8";
* {
    margin: 0;
    padding: 0;
}
#wrap {
    position: relative;
    width: 100vw;
    height: 100vh;
    min-width: 1240px;
    min-height: 500px;
    background-color: antiquewhite;
}
#header {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 80px;
    line-height: 80px;
    text-align: center;
    background-color: violet;
}
#container {
    box-sizing: border-box;
    height: 100vh;
    padding-top: 80px;
    padding-bottom: 50px;
    background-color: coral;
}
.contList {
    height: 100%;
    overflow: hidden;
    background-color: lightblue;
}
.contPager {
    position: absolute;
    right: 50px;
    top: 50%;
    transform: translateY(-50%);
    z-index: 1;
}
.contPager span {
    position: relative;
    display: block;
    width: 15px;
    height: 15px;
    border: 5px solid red;
}
.contPager span::before {
    position: absolute;
    right: 25px;
    top: 0;
    width: 150px;
    text-align: right;
    display: none;
}
.contPager span:nth-child(1)::before {
    content: "섹션 1번 제목";
}
.contPager span:nth-child(2)::before {
    content: "섹션 2번 제목";
}
.contPager span:nth-child(3)::before {

```

```

        content: "섹션 3번 제목";
    }
    .contPager .swiper-pagination-bullet-active {
        background-color: red;
    }

    .contPager .swiper-pagination-bullet-active::before {
        display: block;
    }

    #footer {
        position: absolute;
        bottom: 0;
        left: 0;
        width: 100%;
        height: 50px;
        line-height: 50px;
        text-align: center;
        background-color: violet;
    }
}

```

- **Ex10-2.js**

```

const layOut = new Swiper('.contList', {
    // 세로방향 슬라이드로 움직이게 만듬
    direction: 'vertical',
    pagination: {
        el: '.contPager',
        clickable: true
    },
    // 마우스 휠
    mousewheel: {}
})

```

11. 음원차트 클론 코딩 + 이벤트

- **html #1 (header, top-btn)**

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>음원 차트</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="wrap">
        <div class="header">
            <h1 class="logo"><a href="#"></a></h1>
            <ul class="mainnav">
                <li><a href="#">인기차트</a></li>
                <li><a href="#">추천음악</a></li>
                <li><a href="#">뮤직비디오</a></li>
            </ul>
            <div class="search">
                <input type="search" placeholder="오늘은 무슨 음악을 들을까" >
                
            </div>
            <!-- search_end -->
            <ul class="side_menu">
                <li><a href="#">회원가입</a></li>
                <li><a href="#">로그인</a></li>
            </ul>
        </div>
        <!-- header_end -->

        <!-- main_source (main - 01 ~ 03 붙여넣기)-->

    </div>
    <!-- wrap_end -->
    <button class="top-btn">
        <a href="#"></a>
    </button>
    <!-- button_end -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+3OJU5yExlq6GSYGSKh7tPXikynS7ogEvDej/m4=" crossorigin=>
    <script src="js/music.js"></script>
</body>
</html>

```

- html #2 (main - 01)

```
<div class="main">
    <div class="famous_search">
        <h2>인기검색어</h2>
        <ul>
            <li><a href="#" class="top_number">01</a></li>
            <li><a href="#" class="name">아이유 (IU)</a></li>
        </ul>
        <ul>
            <li><a href="#" class="top_number">02</a></li>
            <li><a href="#" class="name">방탄소년단 (BTS)</a></li>
        </ul>
        <ul>
            <li><a href="#" class="top_number">03</a></li>
            <li><a href="#" class="name">블랙핑크 (BLACKPINK)</a></li>
        </ul>
        <ul>
            <li><a href="#" class="number">04</a></li>
            <li><a href="#">김민석 (멜로망스)</a></li>
        </ul>
        <ul>
            <li><a href="#" class="number">05</a></li>
            <li><a href="#">SOKODOMO</a></li>
        </ul>
        <ul>
            <li><a href="#" class="number">06</a></li>
            <li><a href="#">임영웅</a></li>
        </ul>
        <ul>
            <li><a href="#" class="number">07</a></li>
            <li><a href="#">IVE (아이브)</a></li>
        </ul>
        <ul>
            <li><a href="#" class="number">08</a></li>
            <li><a href="#">이무진</a></li>
        </ul>
        <ul>
            <li><a href="#" class="number">09</a></li>
            <li><a href="#">먼데이 키즈</a></li>
        </ul>
        <ul>
            <li><a href="#" class="number">10</a></li>
            <li><a href="#">비오 (BE'0)</a></li>
        </ul>
    </div>
<!-- famous_search_end -->
```

- html #3 (main -02)

```
<div class="main_img">
    <ul>
        <li class="play">
            <a href="#">
                
                <div class="playlist_wrap">
                    <div class="playlist">
                        <span>수면</span>
                        
                        <p>꿀잠 자게 해줄 잔잔한 음악<br>플레이 리스트</p>
                    </div>
                    <div class="hashtag">
                        <span>#오늘의신곡</span>
                        <span>#꿀잠</span>
                        <span>#수면유도</span>
                        <span>#불면증</span>
                        <span>#낮잠</span><br>
                        <span>#힐링</span>
                        <span>#근吠</span>
                    </div>
                    <div class="count">
                        <a href="#">
                            
                            <span>108</span>
                            
                            <span>15곡</span>
                        </a>
                    </div>
                </div>
            </a>
        </li>
    </ul>
```

```

<li class="play">
    <a href="#">
        
        <div class="playlist_wrap">
            <div class="playlist">
                <span>여행</span>
                
                <p>어디론가 떠나고 싶을 때 듣<br>는 트렌디한 음악</p>
            </div>
            <div class="hashtag">
                <span>#오늘의신곡</span>
                <span>#힐링</span>
                <span>#여행</span>
                <span>#Life</span>
            </div>
            <div class="count">
                <a href="#">
                    
                    <span>76</span>
                    
                    <span>15곡</span>
                </a>
            </div>
        </div>
    </a>
</li>
<li class="play">
    <a href="#">
        
        <div class="playlist_wrap">
            <div class="playlist">
                <span>힙합</span>
                
                <p>그때 그 시절, 90년대 미국<br>인기 힙합 플레이리스트</p>
            </div>
            <div class="hashtag">
                <span>#오늘의신곡</span>
                <span>#힙합</span>
                <span>#90년대</span>
                <span>#HipHop</span>
            </div>
            <div class="count">
                <a href="#">
                    
                    <span>79</span>
                    
                    <span>30곡</span>
                </a>
            </div>
        </div>
    </a>
</li>
<li class="play">
    <a href="#">
        
        <div class="playlist_wrap">
            <div class="playlist">
                <span>일렉트로</span>
                
                <p>스트레스를 한 방에 날려줄<br>신나는 일렉트로댄스 음악</p>
            </div>
            <div class="hashtag">
                <span>#일렉트로</span>
                <span>#일렉트로댄스</span>
                <span>#클럽</span>
                <span>#전자음악</span>
                <span>#댄스</span>
                <span>#스트레스</span>
            </div>
            <div class="count">
                <a href="#">
                    
                    <span>92</span>
                    
                    <span>20곡</span>
                </a>
            </div>
        </div>
    </a>
</li>

```

- **html #4 (main - 03)**

```

<li class="play">
    <a href="#">
        
        <div class="playlist_wrap">
            <div class="playlist">
                <span>재즈</span>
                
                <p>여유로운 카페 음악</p>
            </div>
            <div class="hashtag">
                <span>#공부</span>
                <span>#집중</span>
                <span>#카페음악</span>
                <span>#보사노바</span>
                <span>#재즈</span>
                <span>#힐링</span><br>
                <span>#휴식</span>
            </div>
            <div class="count">
                <a href="#">
                    
                    <span>78</span>
                    
                    <span>15곡</span>
                </a>
            </div>
        </div>
    </a>
</li>
<li class="play">
    <a href="#">
        
        <div class="playlist_wrap">
            <div class="playlist">
                <span>개诘음악</span>
                
                <p>추운 겨울, 당신의 마음을 녹<br>여줄 따뜻한 음악</p>
            </div>
            <div class="hashtag">
                <span>#겨울음악</span>
                <span>#겨울</span>
                <span>#날씨</span>
                <span>#휴식</span>
                <span>#집</span>
            </div>
            <div class="count">
                <a href="#">
                    
                    <span>46</span>
                    
                    <span>15곡</span>
                </a>
            </div>
        </div>
    </a>
</li>
<li class="play">
    <a href="#">
        
        <div class="playlist_wrap">
            <div class="playlist">
                <span>하우스 EDM</span>
                
                <p>기분 좋은 연말, 디너파티를<br>위한 일렉트로 하우스 음악</p>
            </div>
            <div class="hashtag">
                <span>#하우스</span>
                <span>#파티</span>
                <span>#연말</span>
                <span>#크리스마스</span>
                <span>#디너파티</span>
            </div>
            <div class="count">
                <a href="#">
                    
                    <span>118</span>
                    
                    <span>15곡</span>
                </a>
            </div>
        </div>
    </a>
</li>

```

```
</div>
</div>
```

- **style.css**

```
@charset "UTF-8";

*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-size: 14px;
    color: #333;
}

li{
    list-style: none;
}

a{
    text-decoration: none;
}

img{
    display: block;
    border: none;
}
/* reset */

body{
    width: 100%;
    height: 100vh;
}

.wrap{
    width: 100%;
    min-width: 1200px;
    min-height: 800px;
    padding: 0 20px;
}

.header{
    width: 100%;
    height: 80px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    box-shadow: 0px 10px 5px #eee;
}

.mainnav{
    width: 250px;
    display: flex;
    justify-content: space-between;
}

.mainnav a{
    font-weight: bold;
}

.search{
    width: 350px;
    height: 30px;
    display: flex;
    position: relative;
}

.search input{
    width: 100%;
    border-radius: 20px;
    border: 2px solid skyblue;
    padding: 0 30px;
    font-size: 12px;
    line-height: 30px;
}

.search img{
    position: absolute;
    top: 5px;
    left: 5px;
}

.side_menu{
    width: 150px;
```

```

        display: flex;
        justify-content: space-between;
        margin-right: 100px;
        font-weight: bold;
        opacity: .5;
    }
    /* header_end */

    .main{
        margin-top: 30px;
        width: 100%;
        height: 100%;
        display: flex;
        justify-content: space-between;
        padding: 0 40px;
    }

    .famous_search{
        width: 200px;
        height: 100%;
    }

    .famous_search h2{
        padding: 10px 0;
    }

    .famous_search ul{
        display: flex;
        gap: 10px;
        padding-bottom: 15px;
    }

    .famous_search ul .number{
        border: 1px solid #dadada;
        padding: 2px;
        background-color: #dadada;
        border-radius: 3px;
        color: #f5f5f5;
        font-weight: bold;
    }

    .famous_search ul .top{
        background-color: #509ACF;
    }

    .famous_search ul .name{
        font-weight: bold;
    }
    /* famous_search_end */

    .main_img{
        width: calc(100% - 250px);
        height: 100%;
    }

    .main_img > ul > .play{
        display: flex;
        gap: 20px;
        margin-bottom: 30px;
        opacity: 0;
    }

    .playlist_wrap{
        width: 280px;
        /* background-color: gold; */
        padding: 30px 0;
    }

    .playlist{
        width: 100%;
        height: 60px;
        display: flex;
        gap: 20px;
        align-items: center;
    }

    .playlist > span{
        width: 100px;
        position: absolute;
        top: 0;
        left: -380px;
        color: #fff;
        font-weight: bold;
        font-size: 16px;
        border-bottom: 2px solid #fff;
        text-align: center;
        text-shadow: 2px 2px 2px rgba(0,0,0,.8);
    }

```

```

        padding-bottom: 5px;
    }

    .playlist p{
        font-size: 14px;
    }
    /* playlist_end */

    .hashtag{
        padding: 15px 0;
        opacity: .7;
        border-bottom: 2px solid #ccc;
    }

    .hashtag > span{
        font-size: 12px;
    }
    /* hashtag_end */

    .count a{
        padding: 5px 0;
        display: flex;
        gap: 10px;
    }
    /* main_end */

    .top-btn{
        position: absolute;
        bottom: -650px;
        right: 30px;
        background-color: #fff;
        border: none;
    }
}

```

- 스크롤 이벤트 위치값 확인



스트레스를 한 방에 날려줄
신나는 일렉트로댄스 음악

#일렉트로 #일렉트로댄스 #클럽 #전자음악 #댄스
#스트레스

 92 20곡

Default levels ▾ 1 Issue: 1

> \$(window).scrollTop();
< 0
> \$(window).scrollTop();
< 400
> \$(window).scrollTop();
< 758.6666870117188
> \$(window).height();
< 1261
> \$(window).scrollTop() +
\$(window).height();
< 2019.6666870117188

```

// 스크롤 이동에 따른 위치값 확인
$(window).scrollTop();

// 현재 브라우저의 전체 높이 확인
$(window).height();

// 현재 브라우저의 가장 아래의 위치 값 확인(현재 브라우저의 전체 높이 + 스크롤을 가장 아래로 내렸을 때의 위치값)
let windowBottom = $(window).scrollTop() + $(window).height();
console.log(windowBottom);

```

- 요소의 크기값 구성 방식 및 측정 방법

⇒ **div** 요소가 있다고 하면 내부의 여백 `padding` 이 있고 테두리선 `border` 가 있고 외부의 여백인 `margin` 으로 구성되어 있으며, 이 3개가 크기 값에 영향을 미치게 된다.

⇒ `jquery.height()` : `내부높이`

⇒ `jquery.innerHeight()` : `내부높이 + padding`

⇒ `jquery.outerHeight()` : `내부높이 + padding + border`

⇒ **jquery.outerHeight(true)** : 내부높이 + padding + border + margin
 ⇒ **jquery.width()** : 내부넓이
 ⇒ **jquery.innerWidth()** : 내부넓이 + padding
 ⇒ **jquery.outerWidth()** : 내부넓이 + padding + border
 ⇒ **jquery.outerWidth(true)** : 내부넓이 + padding + border + margin



- **music.js**

```

// 전체 playlist의 opacity를 1.5초에 걸쳐 0 -> 1로 바꿔주면서 서서히 나타나게 하는 효과
// $('.play').animate({'opacity' : 1}, 1500)

// 현재 브라우저의 가장 아래의 위치 값 확인(현재 브라우저의 전체 높이 + 스크롤을 가장 아래로 내렸을 때의 위치값)
function scrollHandler(){
  let windowBottom = $(window).scrollTop() + $(window).height();
  // console.log(windowBottom);

  $('.play').each(function(){
    // console.log(this);
    let playList = $(this);

    // playlist 절반값 = 특정 playlist의 위에 있는 좌표 + playlist의 높이값
    let playlistHalf = playList.position().top + playList.outerHeight();

    // playlist 절반값이 현재 브라우저의 가장 아래의 위치 값보다 작은 경우
    // 즉, 스크롤이 전체 높이의 절반에 해당되지 않을 경우 1.5초에 걸쳐 playlist가 나타나도록 처리
    if(playlistHalf < windowBottom){
      playList.animate({'opacity' : '1'}, 1500)
    }
  });

  // 현재 브라우저의 가장 아래의 위치 값과 화면의 높이값이 같을 경우
  // 즉, 스크롤이 가장아래에 있을 경우 top버튼이 보여지도록 하고, 스크롤을 위로 옮리면 숨겨지도록 처리
  if(windowBottom == $(document).height()){
    $('.top-btn').fadeIn();
  } else{
    $('.top-btn').fadeOut();
  }
}

$(window).on('scroll', scrollHandler);

// top 버튼 클릭 시 1초에 걸쳐 최상단으로 스크롤 이동
$('.top-btn').on('click', function(){
  $('html , body').animate({scrollTop : 0}, 1000)
});
  
```

- **JavaScript ES6 최신문법**

⇒ 모던형 자바스크립트

⇒ ES6? - ECMAScript의 약자. Javascript의 표준, 규격을 나타내는 용어. 뒤의 숫자는 버전(자동차로 본다면 모델명).

⇒ 모든 언어는 업데이트를 한다. 처음부터 완벽하게 나오지 않기 때문이다. 그 중에서도 가장 최신 / 이슈가 되었던 업데이트가 ES6이다.

1. 템플릿 리터럴

- 백틱(`)으로 사용
- \${}, 중괄호 앞에 달려 사인을 통해 자바스크립트 표현식의 사용이 가능
- 비교

```
- ES5
var str1 = ', ';
var str2 = 'World!';
var str3 = 'Hello' + str1 + str2;

- ES6
let str1 = ', ';
let str2 = 'World!';
let str3 = `Hello ${str1} ${str2}`;
```

- "" 또는 "를 사용해서 할때에는 연산자를 붙여 넣거나 하는것에 불편한 점이 많았으나 ``과 \${}를 사용해서 간결하게 표현이 가능해졌다.
- 기존까지 사용되던 ""와 "는 동적인값을 표현해 넣기에는 제한사항이 많았으나 ``을 사용하면 \${}를 사용해서 동적인 값을 표현하는 것 이 쉬워졌다.

```
var str1 = ', '
var str2 = 'World!'
var str3 = 'Hello' + str1 + str2
console.log(str3)

console.log("")

let str4 = ', '
let str5 = "World!"
let str6 = `Hello${str1}${str2}`
console.log(str6)
```

2. 객체 리터럴

- 이전 문법보다 훨씬 간결해지 코드로 객체를 선언할 수 있다.
- 메소드에 더이상 클론(:)이나 function을 붙이지 않아도 된다.
- 함수명이 겹치는 경우에는 한 번만 쓸 수 있다.
- 객체의 프로퍼티를 동적으로 생성하려면 객체 리터럴 바깥에서 선언했어야 했는데, ES6 부터는 객체 안에서 바로 속성으로 사용이 가능하다.
- 예시

```
const myFn = function() {
    console.log('myFn');
};

const text = 'TEXT';
const obj = {
    inside() {
        console.log('객체 안에서 바로 함수를 선언한다');
    },
    myFn,
    [text + 1] : '웹컴바인4'
};
obj.inside(); // 출력값 : 객체 안에 바로 함수를 선언한다.
obj.myFn(); // 출력값 : myFn
console.log(obj.text1); // 출력값 : 웹컴바인4
```

3. 구조 / 분해 / 할당

- 펼친다는 뜻으로 객체와 배열에서 사용되어진다.
- 값을 해체한 후 개별 변수에 할당하는 과정을 말한다.

※ 비구조화 할당 예시

```
const person = {
  name: {
    firstName: 'Youngkyun',
    lastName: 'Kim'
  },
  country: 'South Korea',
  gender: 'Male'
}

const counts = [1, 2, 3]

// 기존의 객체, 배열 할당 방식
const firstName = person.name.firstName;
const lastName = person.name.lastName;
const gender = person.gender;
console.log(`${firstName} ${lastName} is a ${gender}`);

const one = counts[0]
const two = counts[1]
const three = counts[2]
console.log(`one + two + three = ${one + two + three}`);

// 비구조화 객체, 배열 할당 문법
const { name: { firstName, lastName }, gender } = person;
console.log(`${firstName} ${lastName} is a ${gender}`);

const [one, two, three] = counts;
console.log(`one + two + three = ${one + two + three}`);

// 비구조화 문법으로 함수 인자에도 적용 가능
const sayHello = ({ name: { firstName, lastName }, country }) => {
  return `Hello, ${firstName} ${lastName} from ${country}!`;
}
console.log(sayHello(person)); // Hello, Youngkyun Kim from South Korea!
```

※ 비구조화 할당을 활용한 RORO 패턴

- **RORO (Receive Object Return Object) 패턴**은 비구조화 할당 문법을 활용한 자바스크립트 프로그래밍 패턴 중 하나입니다. 해당 패턴을 통해 **명명된 매개변수(Named Parameter)**라는 프로그래밍 개념을 유사하게 구현할 수 있으며, 아이디어의 구체적인 내용은 [해당 블로그 포스트](#)에서 확인할 수 있습니다.
- **RORO 패턴의 사용법**

```
// Bad
// 각 전달하는 인자가 어떤 역할을 하는건지 알기 어렵다.
addNewControl("title", 20, 50, 100, 50, true);

// Good
// 인자의 기능을 명확하게 이해할 수 있다.
addNewControl({
  title: "Title",
  xPosition: 20,
  yPosition: 50,
  width: 100,
  height: 50,
  drawingNow: True
})

// 비구조화 할당을 문법을 활용해 구현된 RORO 패턴 함수 선언
function addNewControl({
  title,
  xPosition: xAxis, // alias 가능
  yPosition,
  width=200, // default 값 지정 가능
  height,
  drawingNow
}) {
  // code here
}
```

- **객체 초기화 / 분해 방법**

```

1. 초기화
let name = "JiHun"
let age = 20

- 위의 정보들로 객체를 만드는 방법
[ES5]
let person = 이름 : name , 나이 : age

[ES6]
let person = {name, age}
방식으로 작성이 가능하게 되었다

2. 분해
let girlfriend = {
  name : "여자친구"
  num : 6
}

이러한 구조의 값이 들어있다고 가정할 때, 각각 요소들을 이용하여 새로운 변수를 만드는 방법
[ES5]
let name = girlfriend.name
let num = girlfriend.num

[ES6]
let {name, num} = girlfriend

단, 객체에 사용되어질 키값과 변수의 이름이 같은 경우에만 사용 가능
키 값과 변수의 이름이 다르다면 기존의 방식으로 사용해야 한다

[배열에서의 사용]
let array = [1, 2, 3]

- 위 배열의 내용을 각각의 변수에 담는 방법
[ES5]
let a = array[0]
let b = array[1]
let c = array[2]

[ES6]
let [a, b, c] = array

```

• 사용빈도

- ⇒ 구 문법이 사용되어지는 경우 : 변수의 이름과 객체의 키 값이 다른 경우
- ⇒ 신 문법이 사용되어지는 경우 : 변수의 이름과 객체의 키 값이 같은 경우
- ⇒ 둘 다 빈번하게 사용되어지므로 익숙해져야 한다
- ⇒ 객체뿐 아니라 배열을 대상으로도 사용되어 진다

• 사용빈도가 높은 스프레드 연산자(Spread Operator) : "..."

```

let array = [1, 2, 3]

[배열의 내용을 스프레드 연산자 사용 형태로 변경]
let [a, ...rest] = array
=> a에는 1이 저장되고, 나머지 2, 3은 rest에 저장된다

[객체의 내용을 스프레드 연산자 사용 형태로 변경]
let person = {
  name: "JiHun",
  age : 20,
  cute : true
}

- 위 객체의 내용을 변수로 받아내려면
let {name, age, cute} = person의 방식 -> let {name, ...restInfo} = person의 방식으로 사용 가능
이 때, name에는 "JiHun"이 저장되고, 나머지는 다시 객체로 묶어 restInfo에 저장되어 진다.

- 스프레드 연산자로 할 수 있는 또 한가지
let a = [1, 2]
let b = [3, 4]
let c = [5, 6]

- 이 세개의 배열을 모두 묶고 싶을 때
let result = a.concat(b, c)의 방식 -> let result = [...a, ...b, ...c]의 방식으로 좀 더 직관적으로 사용이 가능하다.

```

- 굉장히 사용빈도가 높은 연산자이다. 특히 **React.js** 또는 **Vue.js** 또는 **Next.js**를 사용한다면 더욱 많이 사용된다.

• 화살표 함수(Arrow function)

⇒ 기존 함수는 **함수식 선언과 변수식 선언, 생성자 함수식 선언** 총 3가지로 구분

```

1. 함수식 선언(선언적 함수)
function 함수명(매개변수...){

}

2. 변수식 선언(익명함수) = 함수표현식
variable 변수명 = function(매개변수...){

}

※ 함수 표현식의 장점
1) 호이스팅 영향 x
2) 클로저로 사용 : 함수를 실행하기 전 해당 함수에 변수를 넘기고 싶을 때 사용
※ 함수 표현식으로 클로저 생성하기
function tabsHandler(index) {
    return function tabClickEvent(event) {
        // 바깥 험수인 tabsHandler() 의 index 인자를 여기서 접근할 수 있다.
        console.log(index); // 탭을 클릭할 때마다 해당 탭의 index 값을 표시
    };
}

var tabs = document.querySelectorAll('.tab');
var i;

for (i = 0; i < tabs.length; i += 1) {
    tabs[i].onclick = tabsHandler(i);
}

3) 콜백으로 사용(다른 함수의 인자로 넘길 수 있음)
// doSth이라는 임시 변수를 사용
var doSth = function () {
    // ...
};

// JQuery 사용 문법과 유사함
$(document).ready(function () {
    console.log('An anonymous function'); // 'An anonymous function'
});

// 자바스크립트 내장 API 인 forEach()를 사용할 때도 콜백함수를 사용할 수 있다.
var arr = ["a", "b", "c"];
arr.forEach(function () {
    // ...
});
★콜백함수 : 다른 함수의 인자로 전달된 함수

3. 생성자 함수식
variable 인스턴스명 = new 생성자함수명(매개변수...);

※ 생성자 함수는 첫 글자는 대문자로 시작, 반드시 new 연산자를 붙여 실행한다는 관례 적용
※ 'new' 를 사용하여 함수 실행 시 아래와 같은 알고리즘 동작
1) 빈 객체를 만들어 this에 할당
2) 함수 본문 실행 -> this에 새로운 프로퍼티를 추가해 this 수정
3) this 반환

※ 재사용할 필요가 없는 복잡한 객체를 만들어야 할 경우 아래와 같이 코드를 익명 생성자 함수로 감싸주는 방식 사용 가능
let user = new function() {
    this.name = "John";
    this.isAdmin = false;

    // 사용자 객체를 만들기 위한 여러 코드.
    // 지역 변수, 복잡한 로직, 구문 등의
    // 다양한 코드가 여기에 삽입
};

4. 화살표 함수
variable 변수명 = (매개변수...) => {

}

※ 매개변수가 없는 경우라도 ()는 사용해 줄 것
※ 화살표 함수에 사용되는 this 키워드 : 함수를 소유한 객체

※ 일반적으로 사용되는 this 키워드의 의미
1) 단독으로 사용 시 '전역객체'를 의미
2) 함수에서는 '전역객체(window)'를 의미
3) 메서드 내부에서는 메서드를 소유한 객체를 의미
4) 이벤트는 이벤트를 받는 대상(객체) 의미

※ 화살표 함수 기본값 문법

```

```
함수명 = (매개변수명 = 기본값...) => {  
}
```

```
5. 즉시호출 함수 : 선언과 호출 동시에 실행하는 함수  
(function(){  
    console.log(3 * 5);  
})();
```

⇒ 함수 선언식은 호이스팅에 영향을 받지만, 함수 표현식은 호이스팅에 영향을 받지 않는다

```
// 실행 전  
logMessage();  
sumNumbers();  
  
function logMessage() {  
    return 'worked';  
}  
  
var sumNumbers = function () {  
    return 10 + 20;  
};  
  
// 실행 시  
function logMessage() {  
    return 'worked';  
}  
  
var sumNumbers;  
  
logMessage(); // 'worked'  
sumNumbers(); // Uncaught TypeError: sumNumbers is not a function  
  
sumNumbers = function () {  
    return 10 + 20;  
};  
  
★★ 호이스팅을 제대로 모르더라도 함수와 변수를 가급적 코드 상단부에서 선언하면, 호이스팅으로 인한 스코프 꼬임 현상은 방지할 수 있다.
```

⇒ 함수 표현식을 화살표 함수로 표현할 수 있다.

⇒ 화살표 함수가 추가되어 함수를 간결하게 나타낼 수 있게 되어 가독성 및 유지 보수성이 올라갔다.

⇒ 만약 함수의 본문에 return만 있는 경우 화살표 함수는 return과 {}를 생략할 수 있다. 단 둘을 같이 생략해야 한다.

⇒ return문에서 소괄호는 사용 가능하다

⇒ ES6에서 가장 크고 중요한 핵심 문법이라고 할 수 있다

```
[ES5]  
function foo(){  
    console.log('곰들이 푸우~')  
}  
  
[ES6]  
let foo = () => {  
    console.log("곰들이 푸우~")  
}  
  
- 큰 차이가 없어보이지만 엄청난 차이를 가지고 있다  
  
[ES5]  
function foo(){  
    return "HAHAHA"  
}  
  
[ES6]  
let foo = () => "HAHAHA"  
- 위 방법처럼 함수의 내용이 한줄 뿐이라면 리턴 키워드의 생략이 가능하고, 중괄호 생략도 가능하다
```

● 주의사항

- 새로운 방식이 생겨났다고 해서 항상 과거의 방식이 완전히 없어지지는 않는다
- 화살표 함수가 편리하긴 하지만 100% 일반 function을 대체할 수 없기 때문이다
- 그 차이점 중 가장 큰 차이점이 this이다

```

let age = 30
let person = {
  name : "JiHun",
  age : 20,
  getInfo : function(){
    console.log(age)
  }
}
person.getInfo()라고 쓴다면 출력 결과는 무엇일까?

```

- this는 나, 나 자신을 가리키는 말이라고 생각하면 쉽다
- 좀 더 정확하게 말하면 함수를 부르는 객체가 바로 this가 되는 것이다
- 일반 함수는 나를 불러낸 객체를 나 자신으로 인식하게끔 생성이 되어지는데, 화살표 함수는 이를 생성하지 못한다
- 화살표 함수가 표현하는 this는 나 자신이 아니라 나를 불러낸 범위가 속해있는 곳을 this로 인식하게 된다
- 이것을 렉시컬 스코프(lexical scope)라고 한다
- 화살표 함수는 선언시기와 호출의 시기도 매우 중요하다
- 호이스팅 대상에 포함되어지지 않는다는 이야기

```

// 기존의 function 문법에서 화살표 함수로 치환하기
function (a) {
  return a + 100;
}

// 1. "function" 키워드를 자우고 인자와 함수 몸통 사이에 화살표 넣기
(a) => {
  return a + 100;
}

// 2. 몸통의 중괄호와 "return" 키워드 지우기 (return 생략)
(a) => a + 100;

// 3. 인자의 꽂호 지우기 (인자가 하나일 경우에만 생략 가능)
a => a + 100;

// 화살함수 사용 예시
const materials = [
  'Hydrogen',
  'Helium',
  'Lithium',
  'Beryllium'
];

// 아래 두 코드는 정확히 같은 문장
console.log(materials.map(material => material.length)); // [8, 6, 7, 9]
console.log(materials.map((material) => { // [8, 6, 7, 9]
  return material.length
}));
```

@@기타 ES6 최신문법

※ 참고 : <https://luckyguystory.tistory.com/101>

1) let, const 키워드

- block scope를 가지고 재선언 불가, 재할당 가능한 let 변수 선언 키워드와 상수 선언 키워드 const가 추가되었음
- 기존에 var 키워드만 있었을 때 보다 예측 가능한 코드를 작성할 수 있게 되었다.

```

// 문제점 1: 기존의 var 변수는 함수를 제외하고 블록 내부에서 선언 되더라도 외부에서 접근 가능
{
  var one = 1;
}
console.log(one); // 1
console.log(window.one); // 1

// 해결: let, const 변수는 블록 외부에서 호출 불가
{
  let two = 2;
  const three = 3;
}
console.log(two); // Uncaught ReferenceError: two is not defined

```

```

console.log(three); // Uncaught ReferenceError: three is not defined

// 문제점 2: var 키워드는 재선언을 허용함
var one = 1;
var one = 1;
var one = 1;
console.log(one); // 1

// 해결: let, const 변수는 재선언 불가능
let two = 2;
let two = 2; // Uncaught SyntaxError: Identifier 'two' has already been declared
const three = 3;
const three = 3; // Uncaught SyntaxError: Identifier 'three' has already been declared

// let vs const 차이점
// let 변수는 새로운 값을 할당할 수 있지만 const 변수는 선언되는 시점에서 할당된 값을 바꿀 수 없음
let two = 2;
two = '2';
const three = 3;
three = '3'; // Uncaught TypeError: Assignment to constant variable

```

2) Class

- Javascript는 프로토타입 기반의 객체 지향 언어이다
- 클래스 기반의 객체 지향 프로그래밍 할 수 있도록 Class 키워드를 도입했다.
- Javascript에서 클래스는 내부적으로 프로토타입을 이용해서 만들어졌다.
- 클래스는 사실 특별한 함수라고 생각해도 좋다.
- 클래스는 호이스팅이 let, const 키워드와 같이 동작한다.

```

class Car{
    constructor(make, color){
        this.make = make;
        this.color = color;
    }
}
let hyundai = new Car("JiHun", "Navy")
car.prototype.summary = function () {
    console.log(`이 자동차는 ${this.make}이 만들었고, 색상은 ${this.color}입니다.`)

    * 당연히 this의 주체가 다르므로 ... 화살표 함수를 사용할 수 없다

    // 기존의 프로토타입 문법
    function Circle (radius) {
        this.radius = radius;
    }

    Circle.prototype = {
        area: function () {
            return this.radius * this.radius * Math.PI;
        }
    }

    console.log(new Circle(2).area()); // 12.566370614359172

    // 클래스 문법
    class Rectangle {
        constructor(height, width) {
            this.height = height;
            this.width = width;
        }
        area() {
            return this.height * this.width;
        }
    };
}

console.log(new Rectangle(5, 8).area()); // 40

```

3) Module

- 모듈이란 재사용하기 위한 코드의 조각을 뜻한다.
- 세부사항은 캡슐화시키고, API 부분만 외부에 노출시킨 코드이다.

- type에 module을 추가시키고 확장자를 mjs로 변경하여 사용
- 모듈은 모듈 스코프를 가지며, import와 export 키워드를 이용해서 사용한다.

```
<script type="module" src="lib.mjs"></script>
```

4) Promise

- Javascript에서 비동기 처리를 기준에는 콜백함수를 사용한 콜백 패턴을 사용했었다.
- 콜백 함수란, 파라미터로 함수를 전달하는 함수, 파라미터로 함수를 전달 받아서 함수의 내부에서 실행하는 함수

```
function add(a, b) {
    return a + b;
};

function sayResult(value) {
    console.log(value);
};

sayResult(add(3, 4));

위의 코드를 콜백함수를 사용한 코드를 바꾸면

function add(a, b, callback) {
    callback(a + b)
};

function sayResult(value) {
    console.log(value)
};

add(3, 4, sayResult)
```

* 콜백 패턴이란, 파라미터가 인자로 전달되는 함수 뒤에 ()를 붙이지 않고 전달해서 실행되는 타이밍을 콜백함수 내부로 옮기는 것. 그러한 패턴을 의미한다.
 - 기존의 자바스크립트에서는 비동기 흐름을 컨트롤하기 위해 이벤트 혹은 콜백 개념을 사용했습니다.
 하지만, 콜백의 사용은 콜백 지옥 (callback hell)이라고 불리는 문제점을 발생시켰으며 이로 인해 복잡성이 증가하고 가독성이 떨어지는 코드가 만들어지곤 했습니다.

결과적으로 callback hell을 발생시킨다.
 - 코드가 중첩되어 복잡함을 야기한다
`step1(function(value1) {
 step2(function(value2) {
 step3(function(value3) {
 step4(function(value4) {
 step5(function(value5) {
 // value6가 파라미터인 또 다른 어떤 함수들..
 })
 })
 })
 })
})`

- 이를 해결하기 위해서 promise가 도입되었다

★ 프로미스 체이닝

```
- 콜백 지옥 예제
// API 호출을 위한 wrapper 함수
function request(url, callback) {
    let xhr = new XMLHttpRequest();
    xhr.open('GET', url);
    xhr.onload = function() {
        callback(JSON.parse(xhr.response));
    };
    xhr.send();
}

const baseUrl = 'https://jsonplaceholder.typicode.com'

// 1. 유저 정보 가져오기
request(`${baseUrl}/users/1`, (user) => {
    // 2. 유저의 포스트 정보 가져오기
    request(`${baseUrl}/posts?userId=${user.id}`, (posts) => {
        // 3. 포스트의 코멘트 정보 가져오기
        request(`${baseUrl}/comments?postId=${posts[0].id}`, (comments) => {
```

```

    // 4. 최종 코드 수행
    console.log(`comments length: ${comments.length}`)
  })
}

// API 호출을 위한 wrapper 함수
function request(url, callback) {
  let xhr = new XMLHttpRequest();
  xhr.open('GET', url);
  xhr.onload = function() {
    callback(JSON.parse(xhr.response));
  };
  xhr.send();
}

// Promise wrapper 함수
function requestPromise(url) {
  return new Promise((resolve, reject) => {
    request(url, resolve)
  })
}

const baseUrl = 'https://jsonplaceholder.typicode.com'

requestPromise(`${baseUrl}/users/1`)
  .then(user => requestPromise(`${baseUrl}/posts?userId=${user.id}`))
  .then(posts => requestPromise(`${baseUrl}/comments?postId=${posts[0].id}`))
  .then(comments => console.log(`comments length: ${comments.length}`))
  .catch(err => console.error(err))

```

- Promise 후속처리 메소드를 이용해서 에러 처리를 효과적으로 할 수 있게 되었다
- Promise는 Javascript 비동기에 사용되는 객체이다.

※ 비동기 처리란, 특정 코드의 실행이 완료될 때 까지 기다리지 않고 다음 코드를 먼저 실행하는 것

```

1. generator function
- function* 선언 (끝에 별표가 있는 function keyword) 은 generator function 을 정의하는데, 이 함수는 Generator 객체를 반환합니다.
- generator function은 GeneratorFunction 생성자와 function* expression 을 사용해서 정의할 수 있습니다.
- 문법
function* name([param[, param[, ... param]]]) {
  statements
}

name : 함수명 , statements : 함수의 본체를 구성하는 구문

Generator는 빠져나갔다가 나중에 다시 돌아올 수 있는 함수입니다. 이때 컨텍스트(변수 값)는 출입 과정에서 저장된 상태로 남아 있습니다.
Generator 함수는 호출되어도 즉시 실행되지 않고, 대신 함수를 위한 Iterator 객체가 반환됩니다. Iterator의 next() 메서드를 호출하면 Generator 함수가 실행되어 yield 이후 next() 메서드가 호출되면 진행이 멈췄던 위치에서부터 재실행합니다. next() 가 반환하는 객체는 yield문이 반환할 값(yielded value)을 나타내는 value 속성과, Ge

- 예시
function* idMaker(){
  var index = 0;
  while(index < 3)
    yield index++;
}

var gen = idMaker();

console.log(gen.next().value); // 0
console.log(gen.next().value); // 1
console.log(gen.next().value); // 2
console.log(gen.next().value); // undefined

- yield* 를 사용한 예시
function* anotherGenerator(i) {
  yield i + 1;
  yield i + 2;
  yield i + 3;
}

function* generator(i){
  yield i;
  yield* anotherGenerator(i);
  yield i + 10;
}

var gen = generator(10);

console.log(gen.next().value); // 10
console.log(gen.next().value); // 11
console.log(gen.next().value); // 12

```

```

console.log(gen.next().value); // 13
console.log(gen.next().value); // 20

- Generator에 인자값을 넘기는 예시
function* logGenerator() {
  console.log(yield);
  console.log(yield);
  console.log(yield);
}

var gen = logGenerator();

// the first call of next executes from the start of the function
// until the first yield statement
gen.next();
gen.next('pretzel'); // pretzel
gen.next('california'); // california
gen.next('mayonnaise'); // mayonnaise

※ Generator는 생성자로서 사용될 수 없다

function* f() {}
  var obj = new f; // throws "TypeError: f is not a constructor"

```

2. async function

async function 선언은 AsyncFunction 객체를 반환하는 하나의 비동기 함수를 정의합니다. 비동기 함수는 이벤트 루프를 통해 비동기적으로 작동하는 함수로, 임시적으로 Promise를 사용하여 결과를 반환합니다. 그러나 비동기 함수를 사용하는 코드의 구문과 구조는, 표준 동기 함수를 사용하는 것과 많이 비슷합니다.

또한 async function expression을 사용해서 async function을 선언할 수 있습니다.

- 문법

```

async function name([param[, param[, ... param]]]) {
  statements
}

```
- 예시

```

function resolveAfter2Seconds() {
  return new Promise(resolve => {
    setTimeout(() => {
      resolve('resolved');
    }, 2000);
  });
}

async function asyncCall() {
  console.log('calling');
  const result = await resolveAfter2Seconds();
  console.log(result);
  // Expected output: "resolved"
}

asyncCall();

```

async 함수에는 await 식이 포함될 수 있습니다. 이 식은 async 함수의 실행을 일시 중지하고 전달 된 Promise의 해결을 기다린 다음 async 함수의 실행을 다시 시작하고 완료된 값을 반환합니다. await 키워드는 async 함수에서만 유효하다는 것을 기억하십시오. async 함수의 본문 외부에서 사용하면 SyntaxError가 발생합니다.

async / await 함수의 목적은 사용하는 여러 promise의 동작을 동기처럼 사용할 수 있게 하고, 어떠한 동작을 여러 promise의 그룹에서 간단하게 동작하게 하는 것이다. promise는 항상 promise를 반환합니다. 만약 async 함수의 반환값이 명시적으로 promise가 아니라면 암묵적으로 promise로 감싸집니다

```

async function foo() {
  return 1
}

function foo() {
  return Promise.resolve(1)
}

- Ex
var resolveAfter2Seconds = function() {
  console.log("starting slow promise");
  return new Promise(resolve => {
    setTimeout(function() {
      resolve(20);
      console.log("slow promise is done");
    }, 2000);
  });
};

var resolveAfter1Second = function() {
  console.log("starting fast promise");
  return new Promise(resolve => {
    setTimeout(function() {
      resolve(10);
    }, 1000);
  });
};

```

```

        console.log("fast promise is done");
    }, 1000);
});
};

var sequentialStart = async function() {
console.log('==SEQUENTIAL START==');

// If the value of the expression following the await operator is not a Promise, it's converted to a resolved Promise.
const slow = await resolveAfter2Seconds();
console.log(slow);

const fast = await resolveAfter1Second();
console.log(fast);
}

var concurrentStart = async function() {
console.log('==CONCURRENT START with await==');
const slow = resolveAfter2Seconds(); // starts timer immediately
const fast = resolveAfter1Second();

console.log(await slow);
console.log(await fast); // waits for slow to finish, even though fast is already done!
}

var stillConcurrent = function() {
console.log('==CONCURRENT START with Promise.all==');
Promise.all([resolveAfter2Seconds(), resolveAfter1Second()]).then((messages) => {
    console.log(messages[0]); // slow
    console.log(messages[1]); // fast
});
}

var parallel = function() {
console.log('==PARALLEL with Promise.then==');
resolveAfter2Seconds().then((message)=>console.log(message));
resolveAfter1Second().then((message)=>console.log(message));
}

sequentialStart(); // after 2 seconds, logs "slow", then after 1 more second, "fast"
// wait above to finish
setTimeout(concurrentStart, 4000); // after 2 seconds, logs "slow" and then "fast"
// wait again
setTimeout(stillConcurrent, 7000); // same as concurrentStart
// wait again
setTimeout(parallel, 10000); // trully parallel: after 1 second, logs "fast", then after 1 more second, "slow"

- async 함수를 사용한 promise chain 재작성
function getProcessedData(url) {
    return downloadData(url) // returns a promise
    .catch(e => {
        return downloadFallbackData(url) // returns a promise
    })
    .then(v => {
        return processDataInWorker(v); // returns a promise
    });
}

async function getProcessedData(url) {
    let v;
    try {
        v = await downloadData(url);
    } catch (e) {
        v = await downloadFallbackData(url);
    }
    return processDataInWorker(v);
}

```

• Promise의 세 가지 상태

- ⇒ Pending(대기) : 비동기 처리로직이 아직 완료되지 않은 상태
- ⇒ Fulfilled(이행) : 비동기 처리가 완료되어 Promise가 결과값을 반환해준 상태
- ⇒ Rejected(실패) : 비동기 처리가 실패하거나 오류가 발생한 상태

• 동기/비동기 예시 코드

```

function printWithDelay(callback, sec) {
    setTimeout(callback, sec + 1000)
}

```

```

printWithDelay(() => console.log('async callback'), 5)
console.log('hello everybody!')

function printImmediately(callBackFunction) {
    callBackFunction()
}

printImmediately(() => console.log('synchronous callback'))
console.log('foo bar asdf!')

```

5) 배열 함수

- **for 루프** : 반복을 위한 문법. 배열과 자주 같이 사용되어진다. 이에 관련된 새로운 문법

```

let names = [
    "Steven Paul",
    "Bill Gates",
    "Mark Elliot Zuckerberg",
    "Elon Musk",
    "Jeff Bezos",
    "Warren Edward Buffett",
    "Larry Page",
    "Larry Ellison",
    "Tim Cook",
    "Lloyd Blankfein"
]

for(let i = 0; i < names.length; i++) {
    console.log(names[i])
}

```

- **forEach(함수)** : 함수를 매개변수로 받는 함수

⇒ 반환값이 없고 단순 for문과 같이 동작함

```

names.forEach(printName);
function printName(item) {
    console.log(item)
};

names.forEach((item) => {console.log(item)})
names.forEach((item, index) => {console.log(item, index)})
names.forEach((item, index, array) => {console.log(item, index, array)})

위와 같이 첫번째 인자는 배열의 각 요소, 두 번째 인자는 배열의 순서값, 세 번째 인자는 배열 자체를 매개변수로 받음

```

- **map(함수)**

⇒ 반환값을 배열에 담아 반환

```

let data = names.map((item) => {
    return item
})

- 첫 번째 인자는 요소의 값, 두 번째 인자는 순서값, 세 번째 인자는 순회하는 대상

let ceoList = [
    {name: "Steven Paul", age: 23, ceo: true},
    {name: "Bill Gates", age: 45, ceo: true},
    {name: "Mark Elliot Zuckerberg", age: 67, ceo: false}
]

let dataList = ceoList.map((item) => {
    return item.name
})
console.log(dataList);

```

- **filter()**

⇒ 조건을 충족하는 요소만 배열에 담아 반환

```
let dataList2 = ceoList.filter((item) => {
    return item.age == 23
})
console.log(dataList2);
```

- **some()** : bool type 값을 반환한다

⇒ 조건을 충족하는 요소가 하나라도 있으면 그에 맞춰 bool type으로 값을 반환

```
let data3 = names.some((item) => {
    return item.startsWith("L")
})
```

- **every()**

⇒ 모든 배열의 요소가 조건을 충족하는지 여부에 따라 bool type으로 값을 반환

```
let data4 = names.every((item) => {
    return item.length < 0
})
console.log(data4)
```

- **find()** : 배열이 아닌 string으로 반환, 정확한 하나의 값을 찾기에 적합

⇒ 조건을 충족하는 요소 하나만 반환(여러개라면 첫 번째 것만 반환)

```
let data5 = names.find((item) => {
    return item.startsWith("L")
})
```

- **findIndex()**

⇒ 조건을 충족하는 요소의 순서값을 반환(여러개라면 첫 번째 것만 반환)

```
let data6 = names.findIndex((item) => {
    return item.startsWith("L")
})
```

※ 예시

```
let names = [
    "Steven Paul",
    "Bill Gates",
    "Mark Elliot Zuckerberg",
    "Elon Musk",
    "Jeff Bezos",
    "Warren Edward Buffett",
    "Larry Page",
    "Larry Ellison",
    "Tim Cook",
    "Lloyd Blankfein"
]

let data = names.map((item) => {
    return item
})

let ceoList = [
    {name: "Steven Paul", age: 23, ceo: true},
```

```

        {name: "Bill Gates", age: 45, ceo: true},
        {name: "Mark Elliot Zuckerberg", age: 67, ceo: false}
    ]

let dataList = ceoList.map((item) => {
    return item.name
})
console.log(dataList);

let dataList2 = ceoList.filter((item) => {
    return item.age == 23
})
console.log(dataList2);

let data2 = names.filter((item) => {
    return item.startsWith("L")
})
console.log(data2)

let data4 = names.every((item) => {
    return item.length < script 0
})
console.log(data4)

let data6 = names.findIndex((item) => {
    return item.startsWith("L")
})
console.log(data6)

```

※ 배열함수 예시

```

let names = [
    "Steven Paul Jobs",
    "Bill Gates",
    "Mark Elliot Zuckerberg",
    "Elon Musk",
    "Jeff Bezos",
    "Warren Edward Buffett",
    "Larry Page",
    "Larry Ellison",
    "Tim Cook",
    "Lloyd Blankfein",
];
// map()
// 모든 이름을 대문자로 바꾸어 출력
let capitalNames = names.map((item) => {
    return item.toUpperCase()
})
console.log(capitalNames);

// 성을 제외한 이름만 출력
let firstNames = names.map((item1) => {
    return item1.split(" ")[0]
})
console.log(firstNames);

// 이름의 이니셜만 출력
let initialNames = names.map((item2) => {
    let splittedName = item2.split(" ");
    let initials = "";
    splittedName.forEach((sumInitials) => {
        initials+=sumInitials[0]
    });
    return initials;
})
console.log(initialNames);

// filter()
// 이름에 a를 포함한 사람들을 출력
let withA = names.filter((person1) => {
    return person1.includes('a')
})
console.log(withA);

// 이름에 같은 글자가 연속해서 들어간 사람들을 출력(aa, bb 와 같이)
let doubleLetter = names.filter((person2) => {
    let splittedName = person2.split("")
    return splittedName.some((lett, index) => lett == splittedName[index + 1])
})
console.log(doubleLetter)

```

```

// some()
// 전체 이름의 길이가 20자 이상인 사람이 있는가?
let twentyWords = names.some((person3) => {
    return person3.length >= 20
});
console.log(twentyWords);

// 성을 제외한 이름에 p를 포함한 사람이 있는가(대소문자 무관)
console.log(names.some((item) => {
    let splitName = item.split(" ")
    splitName.pop()
    return splitName.some(eachName => eachName.toLocaleLowerCase().includes('a'))
}));


// every()
// 모두의 전체 이름의 길이가 20자 이상인가?
console.log(names.every((item) => {
    return item.length >= 20
}));


// 모두의 이름에 a가 포함되어 있는가?
console.log(names.every((item) => {
    return item.includes('a')
}));


// find()
// 전체 이름의 길이가 20자 이상인 사람을 찾으시오
console.log(names.find((item) => {
    return item.length >= 20
}));


// 미들 네임이 포함되어 있는 사람을 찾으시오
console.log(names.find((item) => item.split(' ').length >= 3));


// findIndex()
// 전체 이름의 길이가 20자 이상인 사람의 순서값을 찾으시오
console.log(names.findIndex((item) => item.length >= 20));


// 미들네임이 포함되어 있는 사람의 순서값을 찾으시오
console.log(names.findIndex((item) => item.split(" ").length >= 3));

```

6) 트랜스파일링

- 모든 웹브라우저가 ES6를 지원하지는 않고, 지원하더라도 모든 기능을 지원하지 않는 경우가 많다. 그러니 브라우저에서 ES6코드를 실행하기 전에 ES5로 컴파일하면 ES6가 제대로 작동하도록 보장할 수 있다.
- 이런 변환을 **트랜스파일링(transpiling)**이라고 한다.
- 가장 유명한 트랜스파일링 도구로는 바벨(Babel)이 있다.
- 트랜스파일링으로 브라우저가 최신 자바스크립트 기능을 지원할 때 까지 기다리지 않아도 바로 사용할 수 있게 됐다!
- 트랜스파일링은 한 버전의 자바스크립트 코드를 더 많은 브라우저가 이해할 수 있는 다른 버전의 자바스크립트 구문으로 변환하는 것이다.
- 게다가 이제는 자바스크립트에서 소스코드가 생겨났다.
- 이 말은 브라우저에서 직접 실행할 수 있는 파일이 프로젝트에 들어있는 경우도 있다는 뜻이다.

```

const add = (x=5, y=10) => console.log(x+y);

이 코드를 트랜스파일러로 변환하면 다음과 같은 출력이 생긴다.

"use strict";

var add = function add(){
    var x = arguments.length <= 0 || arguments[0] === undefined ? 5 : arguments[0];
    var y = arguments.length <= 1 || arguments[1] === undefined ? 10 : arguments[1];
    return console.log(x+y);
};

트랜스파일러는 use strict 선언을 맨 위에 추가해서 코드가 엄격한 모드에서 실행되도록 만든다.
x와 y 파라미터의 디폴트 값은 arguments 배열로 처리된다.
이렇게 만들어진 자바스크립트는 다양한 브라우저에서 사용 가능하다.

```

인라인 바벨 트랜스파이럴을 사용하면 브라우저에서 자바스크립트를 직접 트랜스파일 할 수도 있다.
 browser.js 파일을 포함시키고(HTML의 script태그 사용), 변환하고 싶은 코드의 script 태그에 type="text/babel"을 지정하면 된다.

```

<div id="output"></div>
<!--바벨 로딩-->
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
<!--변환할 코드를 script태그 안에 넣기 -->
<script type="text/babel">
const getMessage = () => "Hello World";
document.getElementById('output').innerHTML = getMessage();
</script>
<!--파일에 있는 소스 코드를 트랜스파일링 하기-->
<script src="script.js" type="text/babel">

```

[기타 Project]

Ex01. 윷놀이게임

- index.html

```

<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>윷놀이게임</title>
    <link rel="stylesheet" href="yuch.css">
    <script src="jquery-1.12.3.js"></script>
    <script src="yuch.js"></script>
  </head>
  <body>
    <h3>윷놀이게임</h3>
    <div class="yuch_play">
      <div class="yuch01"></div>
      <div class="yuch02"></div>
      <div class="yuch03"></div>
      <div class="yuch04"></div>
    </div>
    <button>윷 돌리기</button>
  </body>
</html>

```

- style.css

```

@charset "utf-8";

*{
  padding: 0;
  margin: 0;
}

h3{
  width: 100%;
  height: 100px;
  text-align: center;
  line-height: 100px;
  font-size: 50px;
  color: #603913;
}

/* 윷판 */
.yuch_play{
  /* 창사이즈가 1200보다 작아지면 100%로 처리 */
  max-width: 100%;
  width: 1200px;
  padding: 100px;
  border: 3px solid #603913;
  box-sizing: border-box;
  margin: 0 auto 20px;
  display: flex;
}

.yuch_play > div{
  width: 25%;
  height: 500px;
  background-image: url(front.png);
  background-repeat: no-repeat;
  background-position: center;
  background-size: contain;
}

button{
  display: block;
}

```

```

width: 250px;
height: 50px;
margin: 0 auto;
border: 3px solid #603913;
box-sizing: border-box;
background-color: #c37e39;
font-size: 20px;
color: #603913;
font-weight: bold;
cursor: pointer;
}

/* 활성처리 #1 - 뒷면 보이게 처리 */
.yuch_play > div.active{
    background-image: url(back.png);
}

.yuch_play > div.yuch04.active{
    background-image: url(back2.png);
}

/* 활성처리 #2 - 낙! */
.yuch_play > div.yuch04.nak{
    position: relative;
    left: 200px;
    transform: rotate(20deg);
}

.yuch_play > div.yuch04.nak:after{
    content: url(double.png);
    position: absolute;
    top: -50px;
    left: -50px;
}

/* 활성 #3 - 애니메이션 처리 */
.yuch_play > div.rotate{
    animation: rotate 0.2s 3;
}

/* rotate 애니메이션 */
@keyframes rotate{
    from{ transform: rotate(0deg); }
    to{ transform: rotate(360deg); }
}

/* 1199 ~ 태블릿까지 */
@media screen and (max-width: 1199px){
    .yuch_play{
        width: calc(100% - 40px);
        padding: 50px; margin: 20px auto;
    }

    .yuch_play > div.yuch04.nak{
        top: -100px;
        left: 0;
    }

    .yuch_play > div.yuch04.nak:after{
        top: 20px;
        left: -300px;
    }
}

/* 700보다 작을 때 처리 */
@media screen and (max-width: 700px){
    h3{
        font-size: 24px;
        height: 50px;
        line-height: 50px;
    }

    .yuch_play{
        padding: 20px;
    }

    .yuch_play > div{
        height: 300px;
    }

    .yuch_play > div.yuch04.nak:after{
        transform: scale(0.7);
        top: 20px; left: -230px;
    }
}

```

- **yuch.js**

```

$(document).ready(function(){

  $('button').click(function(){
    // 0. nak클래스를 처음부터제거
    $('.yuch_play > div.yuch04').removeClass('nak');

    // 0. 클릭하자마자 버튼 사용 불가능 처리
    $(this).prop('disabled',true);

    // 1. rotate클래스가 추가 - 마지막에 삭제
    $('.yuch_play > div').addClass('rotate');

    // 2. rotate 애니메이션 후 랜덤 명령 실행
    setTimeout(function(){
      // 3. 낙처리 - 10번 중 한번 처리
      // - 1~10사이의 랜덤한 정수를 담는 변수
      var nak = Math.floor(Math.random() * 9) + 1;

      if(nak == 7){ // 1~10의 어떤숫자를 써도 됨
        $('.yuch_play > div.yuch04').addClass('nak');
      }

      // 4. 각각의 뒷이 랜덤하게 처리
      var random01, random02, random03, random04;

      // 4개를 한번에 처리하기 위해 반복문
      for(var i=1;i<=4;i++){
        // 각각 변수에 1과 2를 랜덤하게 담기
        eval('random' + i + ' = Math.floor(Math.random() * 2) + 1;');

        // 현재 반복하고 있는 변수 담기
        var ran = eval('random' + i);

        if(ran == 1){ // 1과 같다면 앞면보이게 처리
          $('.yuch_play > div.yuch0' + i).removeClass('active');
        }else{ // 2와 같으면 뒷면이 보이게 처리
          $('.yuch_play > div.yuch0' + i).addClass('active');
        }
      }

      // 5. rotate제거해서 다시 클릭시 애니메이션 적용
      $('.yuch_play > div').removeClass('rotate');

      // 6. 버튼을 다시 사용하게 처리
      $('button').prop('disabled',false);
    },600);
  });
});

```

Ex02. ‘오징어게임’ Intro 화면 만들기

- **index.html**

```

<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8">
    <title>오징어게임 인트로</title>
    <link rel="stylesheet" href="ani.css">
  </head>
  <body>
    <div class="canvas">
      <!-- 상원 -->
      <svg class="c01">
        <circle cx="70" cy="70" r="57.5" />
      </svg>
      <!-- 상원라인 -->
      <div class="c01_line01"></div>
      <div class="c01_line02"></div>
      <div class="c01_line03"></div>
      <div class="c01_line04"></div>

      <!-- 삼각형 -->
      <svg class="tri">
        <polygon points="145,25 268,238 22,238" />
      </svg>
      <div class="black"></div>
    </div>
  </body>

```

```

<svg class="tri02">
  <polygon points="145,25 268,238 22,238" />
</svg>

<!-- 사각형 -->
<div class="rect">
  <span class="line01"></span>
  <span class="line02"></span>
  <span class="line03"></span>
  <span class="line03_1"></span>
  <span class="line04"></span>
</div>

<!-- 하원 -->
<svg class="c02">
  <circle cx="70" cy="70" r="57.5" />
</svg>
<!-- 하원라인 -->
<div class="c02_line01"></div>
<div class="c02_line02"></div>
<div class="c02_line03"></div>
<div class="c02_line04"></div>

<!-- ○ -->
<svg class="c03">
  <circle cx="55" cy="55" r="40" />
</svg>
<!-- 다른색 -->
<svg class="c04">
  <circle cx="55" cy="55" r="40" />
</svg>
<!-- ± -->
<div class="first_letter">
  <span class="line01"></span>
  <span class="line02"></span>
</div>

<!-- × -->
<div class="second_letter01"></div>
<!-- l -->
<div class="second_letter02"></div>
<!-- ○ -->
<svg class="c05">
  <circle cx="42.5" cy="42.5" r="30" />
</svg>

<!-- ○ -->
<svg class="c06">
  <circle cx="42.5" cy="42.5" r="30" />
</svg>
<!-- - -->
<div class="third_letter01"></div>
<!-- l -->
<div class="third_letter02"></div>

<!-- - -->
<div class="fourth_letter01"></div>
<!-- / -->
<div class="fourth_letter02_wrap">
  <div class="fourth_letter02"></div>
</div>
<!-- - -->
<div class="fourth_letter03"></div>
<!-- | -->
<div class="fourth_letter04"></div>
<!-- | -->
<div class="fourth_letter05"></div>

<!-- ○ -->
<svg class="c07">
  <circle cx="42.5" cy="42.5" r="30" />
</svg>
<!-- l -->
<div class="fifth_letter01"></div>
<!-- □ -->
<div class="fifth_letter02"></div>
<div class="fifth_letter04"></div>
<div class="fifth_letter05"></div>
</div>
</body>
</html>

```

- **style.css #1-1**

```

@charset "utf-8";

*{
    padding: 0;
    margin: 0;
}

body{
    background-color: #181818;
}

.canvas{
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    margin: auto;
    width: 760px;
    height: 640px;
}

.c01{
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    margin: 0 auto;
    width: 140px;
    height: 140px;
    z-index: 50;
}

.c02{
    position: absolute;
    bottom: 0;
    left: 0;
    right: 0;
    margin: 0 auto;
    width: 140px;
    height: 140px;
    z-index: 20;
}

.c01 circle, .c02 circle{
    fill: transparent;
    stroke: #e6ebce;
    stroke-width: 25px;
    stroke-dasharray: 361px;
    stroke-dashoffset: 0;
    transform-origin: center;
}

.first_letter{
    width: 165px;
    height: 70px;
    position: absolute;
    left: 0;
    bottom: 300px;
    z-index: 13;
}

.first_letter > .line01{
    width: 165px;
    height: 25px;
    background-color: #e6ebce;
    position: absolute;
    right: 0;
    bottom: 0;
    transform-origin: right center;
    transform: scaleX(0);
    animation: first_line01 2s ease-in-out;
    animation-delay: 1s;
    animation-fill-mode: forwards;
}

.first_letter > .line02{
    width: 25px;
    height: 45px;
    background-color: #e6ebce;
    position: absolute;
    right: 55px;
    bottom: 25px;
    transform-origin: center bottom;
    transform: scaleY(0);
}

```

```

        animation: first_line02 0.5s ease-in-out;
        animation-delay: 5.5s;
        animation-fill-mode: forwards;
    }

    @keyframes first_line01{
        from{
            transform: scaleX(0);
        }

        to{
            transform: scaleX(1);
        }
    }

    @keyframes first_line02{
        from{
            transform: scaleY(0);
        }

        to{
            transform: scaleY(1);
        }
    }

.c02 circle{
    transform: rotate(90deg);
    animation: c02_circle 1s ease-in-out;
    animation-delay: 1s;
    animation-fill-mode: forwards;
}

@keyframes c02_circle{
    from{
        stroke-dashoffset: 0;
    }

    to{
        stroke-dashoffset: -361px;
    }
}

.c01 circle{
    transform: rotate(-90deg);
    animation: c02_circle 1s ease-in-out;
    animation-delay: 1s;
    animation-fill-mode: forwards;
}

@keyframes c01_circle{
    from{
        stroke-dashoffset: 0;
    }

    to{
        stroke-dashoffset: -361px;
    }
}

.c01_line01{
    width: 313px;
    height: 25px;
    background-color: #e6ebce;
    position: absolute;
    top: 0;
    right: 70px;
    transform-origin: left center;
    transform: scaleX(0);
    animation: c02_line01 1s ease-in-out;
    animation-delay: 1s;
    animation-fill-mode: forwards;
    z-index: 15;
}

.c01_line02{
    width: 314px;
    height: 25px;
    background-color: #181818;
    position: absolute;
    top: 0;
    right: 70px;
    transform-origin: left center;
    transform: scaleX(0);
    animation: c02_line02 1s ease-in-out;
    animation-delay: 2s;
    animation-fill-mode: forwards;
    z-index: 16;
}

```

```

}

.c01_line03{
  width: 25px;
  height: 225px;
  background-color: #e6ebce;
  filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
  position: absolute;
  top: 0;
  right: 70px;
  transform-origin: center top;
  transform: scaleY(0);
  animation: c02_line03 1s ease-in-out;
  animation-delay: 2.7s;
  animation-fill-mode: forwards;
  z-index: 17;
}

.c01_line04{
  width: 25px;
  height: 225px;
  background-color: #e6ebce;
  filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
  position: absolute;
  top: 0;
  right: 70px;
  transform-origin: center bottom;
  transform: scaleY(0);
  animation: c02_line04 1s ease-in-out;
  animation-delay: 3.6s;
  animation-fill-mode: forwards;
  z-index: 18;
}

@keyframes c02_line01{
  from{
    transform: scaleX(0);
  }

  to{
    transform: scaleX(1);
  }
}

@keyframes c02_line02{
  from{
    transform: scaleX(0);
  }

  to{
    transform: scaleX(1);
  }
}

@keyframes c02_line03{
  from{
    transform: scaleY(0);
  }

  99%{
    transform: scaleY(1);
    opacity: 1;
  }

  to{
    transform: scaleY(1);
    opacity: 0;
  }
}

@keyframes c02_line04{
  from{
    transform: scaleY(1);
  }

  to{
    transform: scaleY(0);
  }
}

.c02_line01{
  width: 261px;
  height: 25px;
  background-color: #e6ebce;
  position: absolute;
  bottom: 0;
  left: 124px;
  transform-origin: right center;
  transform: scaleX(0);
}

```

```

        animation: c02_line01 is ease-in-out;
        animation-delay: 1s;
        animation-fill-mode: forwards;
        z-index: 15;
    }

.c02_line02{
    width: 262px;
    height: 25px;
    background-color: #181818;
    position: absolute;
    bottom: 0;
    left: 124px;
    transform-origin: right center;
    transform: scaleX(0);
    animation: c02_line02 is ease-in-out;
    animation-delay: 2s;
    animation-fill-mode: forwards;
    z-index: 16;
}

.c02_line03{
    width: 25px;
    height: 420px;
    background-color: #e6ebce;
    filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
    position: absolute;
    bottom: 0;
    left: 124px;
    transform-origin: center bottom;
    transform: scaleY(0);
    animation: c02_line03 is ease-in-out;
    animation-delay: 2.7s;
    animation-fill-mode: forwards;
    z-index: 17;
}

.c02_line04{
    width: 25px;
    height: 420px;
    background-color: #e6ebce;
    filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
    position: absolute;
    bottom: 0;
    left: 124px;
    transform-origin: center top;
    transform: scaleY(0);
    animation: c02_line04 is ease-in-out;
    animation-delay: 3.6s;
    animation-fill-mode: forwards;
    z-index: 18;
}

@keyframes c02_line01{
    from{
        transform: scaleX(0);
    }

    to{
        transform: scaleX(1);
    }
}

@keyframes c02_line02{
    from{
        transform: scaleX(0);
    }

    to{
        transform: scaleX(1);
    }
}

@keyframes c02_line03{
    from{
        transform: scaleY(0);
    }

    99%{
        transform: scaleY(1);
        opacity: 1;
    }

    to{
        transform: scaleY(1);
        opacity: 0;
    }
}

```

```

@keyframes c02_line04{
  from{
    transform: scaleY(1);
  }
  to{
    transform: scaleY(0);
  }
}

.c03{
  position: absolute;
  bottom: 362.5px;
  left: 41.5px;
  width: 110px;
  height: 110px;
  z-index: 40;
}

.c03 circle{
  fill: transparent;
  stroke: #e6ebce;
  stroke-width: 25px;
  stroke-dasharray: 251px;
  stroke-dashoffset: -251px;
  transform: rotate(0deg);
  transform-origin: center;
  animation: c03_circle 1s ease-in-out;
  animation-delay: 3.6s;
  animation-fill-mode: forwards;
}

@keyframes c03_circle{
  from{
    stroke-dashoffset: -251px;
  }
  to{
    stroke-dashoffset: 0;
  }
}

.c04{
  position: absolute;
  bottom: 362.5px;
  left: 41.5px;
  width: 110px;
  height: 110px;
  position: absolute;
}

.c04 circle{
  fill: transparent;
  stroke: #fe4063;
  stroke-width: 25px;
  filter: drop-shadow(0px 5px 10px rgba(0,0,0,0.5));
  stroke-dasharray: 251px;
  stroke-dashoffset: 0px;
  transform: rotate(0deg);
  transform-origin: center;
  opacity: 0;
  animation: c04_circle 0.5s ease-in-out;
  animation-delay: 4.6s;
  animation-fill-mode: forwards;
}

@keyframes c04_circle{
  from{
    opacity: 0;
  }
  to{
    opacity: 1;
  }
}

.tri{
  position: absolute;
  top: 60px;
  left: 0;
  right: 0;
  margin: 0 auto;
  width: 290px;
  height: 250px;
  animation: tri01_parent 2s ease-in-out;
}

```

```

        animation-delay: 3s;
        animation-fill-mode: forwards;
        z-index: 60;
    }

    @keyframes tri01_parent{
        from{
            transform: translateX(0);
        }

        to{
            transform: translateX(-235px);
        }
    }

    .tri polygon{
        fill: transparent;
        stroke: #e6ebce; stroke-width: 25px;
        filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
        transform-origin: calc(100% - 21px) calc(100% - 10px);
        animation: tri01 4s ease-in-out;
        animation-delay: 1s;
        animation-fill-mode: forwards;
    }

    @keyframes tri01{
        from{
            transform: scale(1);
            stroke-width: 25px;
            stroke: #e6ebce;
            filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
        }

        50%{
            transform: scale(0.26);
            stroke-width: 100px;
            stroke: #e6ebce;
            filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
        }

        90%{
            transform: scale(0.26);
            stroke-width: 100px; stroke: #e6ebce;
            filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
        }

        to{
            transform: scale(0.26);
            stroke-width: 100px;
            stroke: #fe4063;
            filter: drop-shadow(2px -5px 10px rgba(0,0,0,0.5));
        }
    }

    .black{
        width: 290px;
        height: 227px;
        position: absolute;
        top: 60px;
        left: 0;
        right: 0;
        margin: 0 auto;
        background-color: #181818;
        z-index: 31;
    }

    .tri02{
        position: absolute;
        top: 60px;
        left: 0;
        right: 0;
        margin: 0 auto;
        width: 290px;
        height: 250px;
        transform-origin: left center;
        animation: tri02 2s ease-in-out;
        animation-delay: 3s;
        animation-fill-mode: forwards;
        z-index: 30;
    }

    @keyframes tri02{
        from{
            transform: scaleX(1);
        }

        to{
    
```

```

        transform: scaleX(0);
    }
}

.tri02 polygon{
    fill: transparent;
    stroke: #e6ebce; stroke-width: 25px;
    filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
}

.second_letter01{
    width: 105px;
    height: 25px;
    background-color: #e6ebce;
    position: absolute;
    left: 185px;
    bottom: 405px;
    transform-origin: left center;
    transform: scaleX(0);
    animation: second_letter01 is ease-in-out;
    animation-delay: 3s;
    animation-fill-mode: forwards;
    z-index: 59;
}

@keyframes second_letter01{
    from{
        transform: scaleX(0);
    }

    to{
        transform: scaleX(1);
    }
}

```

- **style.css #1-2**

```

.second_letter02{
    width: 25px;
    height: 140px;
    background-color: #e6ebce;
    position: absolute;
    left: 320px; bottom: 355px;
    transform-origin: center top;
    transform: scaleY(0);
    animation: second_letter02 is ease-in-out;
    animation-delay: 2s;
    animation-fill-mode: forwards;
    z-index: 58;
}

@keyframes second_letter02{
    from{
        transform: scaleY(0);
    }

    to{
        transform: scaleY(1);
    }
}

.c05{
    position: absolute;
    bottom: 365px;
    bottom: 250px;
    left: 270px;
    width: 85px;
    height: 85px;
    z-index: 57;
}

.c05 circle{
    fill: transparent;
    stroke: #e6ebce;
    stroke-width: 25px;
    filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
    stroke-dasharray: 251px;
    stroke-dashoffset: 0;
    transform: rotate(0deg);
    transform-origin: center;
    transform: scale(0);
    animation: c05_circle is ease-in-out;
    animation-delay: 3s;
    animation-fill-mode: forwards;
}

```

```

@keyframes c05_circle{
  from{
    transform: scale(0);
  }
  to{
    transform: scale(1);
  }
}

.c06{
  position: absolute;
  bottom: 365px;
  bottom: 375px;
  left: 360px;
  width: 85px;
  height: 85px;
  z-index: 57;
}

.c06 circle{
  fill: transparent;
  stroke: #e6ebce;
  stroke-width: 25px;
  filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
  stroke-dasharray: 251px;
  stroke-dashoffset: 0;
  transform: rotate(0deg);
  transform-origin: center;
  transform: scale(0);
  animation: c06_circle 1s ease-in-out;
  animation-delay: 4s;
  animation-fill-mode: forwards;
}

@keyframes c06_circle{
  from{
    transform: scale(0);
  }
  to{
    transform: scale(1);
  }
}

.third_letter01{
  width: 45px;
  height: 25px;
  background-color: #e6ebce;
  position: absolute;
  left: 440px;
  bottom: 405px;
  transform-origin: right center;
  transform: scaleX(0);
  animation: third_letter01 1s ease-in-out;
  animation-delay: 4s;
  animation-fill-mode: forwards;
  z-index: 59;
}

@keyframes third_letter01{
  from{
    transform: scaleX(0);
  }
  to{
    transform: scaleX(1);
  }
}

.third_letter02{
  width: 25px;
  height: 140px;
  background-color: #e6ebce;
  position: absolute;
  left: 485px;
  bottom: 355px;
  transform-origin: center top;
  transform: scaleY(0);
  animation: third_letter02 1s ease-in-out;
  animation-delay: 3s;
  animation-fill-mode: forwards;
  z-index: 58;
}

@keyframes third_letter02{

```

```

        from{
            transform: scaleY(0);
        }

        to{
            transform: scaleY(1);
        }
    }

.fourth_letter01{
    width: 50px;
    height: 25px;
    background-color: #e6ebce;
    position: absolute;
    left: 465px;
    bottom: 315px;
    transform-origin: left center;
    transform: scaleX(0);
    animation: fourth_letter01 0.5s ease-in-out;
    animation-delay: 4s;
    animation-fill-mode: forwards;
    z-index: 70;
}

@keyframes fourth_letter01{
    from{
        transform: scaleX(0);
    }

    to{
        transform: scaleX(1);
    }
}

.fourth_letter02_wrap{
    width: 84px;
    height: 0;
    position: absolute;
    left: 455px; bottom: 215px;
    overflow: hidden;
    animation: fourth_letter02 1s ease-in-out;
    animation-delay: 4.5s;
    animation-fill-mode: forwards;
    z-index: 69;
}

@keyframes fourth_letter02{
    from{
        height: 0;
    }

    to{
        height: 125px;
    }
}

.fourth_letter02{
    width: 25px;
    height: 125px;
    background-color: #e6ebce;
    position: absolute;
    left: 0;
    bottom: 0;
    filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
    transform: skewX(-25deg);
    transform-origin: left bottom;
}

.fourth_letter03{
    width: 50px;
    height: 25px;
    background-color: #e6ebce;
    position: absolute;
    left: 510px;
    bottom: 280px;
    transform-origin: left center;
    transform: scaleX(0);
    animation: fourth_letter03 1s ease-in-out;
    animation-delay: 5.5s;
    animation-fill-mode: forwards;
    z-index: 68;
}

@keyframes fourth_letter03{
    from{
        transform: scaleX(0);
    }
}

```

```

        to{
            transform: scaleX(1);
        }
    }

.fourth_letter04{
    width: 25px;
    height: 100px;
    background-color: #e6ebce;
    position: absolute;
    left: 555px;
    bottom: 240px;
    transform-origin: center bottom;
    transform: scaleY(0);
    animation: fourth_letter04 1s ease-in-out;
    animation-delay: 3s;
    animation-fill-mode: forwards;
    z-index: 69;
}

@keyframes fourth_letter04{
    from{
        transform: scaleY(0);
    }

    to{
        transform: scaleY(1);
    }
}

.fourth_letter05{
    width: 25px;
    height: 160px;
    background-color: #e6ebce;
    position: absolute;
    left: 600px;
    bottom: 180px;
    transform: scaleY(0);
    transform-origin: center bottom;
    animation: fourth_letter05 1s ease-in-out;
    animation-delay: 4s;
    animation-fill-mode: forwards;
    z-index: 69;
}

@keyframes fourth_letter05{
    from{
        transform: scaleY(0);
    }

    to{
        transform: scaleY(1);
    }
}

.c07{
    position: absolute;
    bottom: 365px;
    bottom: 375px;
    left: 605px;
    width: 85px;
    height: 85px;
    z-index: 57;
}

.c07 circle{
    fill: transparent;
    stroke: #e6ebce;
    stroke-width: 25px;
    filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
    stroke-dasharray: 189px;
    stroke-dashoffset: 189px;
    transform: rotate(0);
    transform-origin: center;
    animation: c07_circle 1s ease-in-out;
    animation-delay: 3.6s;
    animation-fill-mode: forwards;
}

@keyframes c07_circle{
    from{
        stroke-dashoffset: 189px;
    }

    to{
        stroke-dashoffset: 0;
    }
}

```

```

}

.fifth_letter01{
  width: 25px;
  height: 140px;
  background-color: #e6ebce;
  position: absolute;
  right: 0px;
  bottom: 355px;
  transform: scaleY(0);
  transform-origin: center top;
  animation: fifth_letter01 1s ease-in-out;
  animation-delay: 5s;
  animation-fill-mode: forwards;
  z-index: 69;
}

@keyframes fifth_letter01{
  from{
    transform: scaleY(0);
  }

  to{
    transform: scaleY(1);
  }
}

.fifth_letter02{
  width: 95px;
  height: 25px;
  background-color: #fe4063;
  position: absolute;
  right: 125px;
  bottom: 315px;
  transform-origin: left center;
  transform: scaleX(0);
  animation: fifth_letter02 2s ease-in-out;
  animation-delay: 3s;
  animation-fill-mode: forwards;
  z-index: 68;
}

@keyframes fifth_letter02{
  from{
    transform: scaleX(0);
    width: 95px;
    transform-origin: left center;
  }

  30%{
    transform: scaleX(1);
    width: 150px;
    transform-origin: left center;
  }

  50%{
    transform: scaleX(1);
    width: 95px;
    transform-origin: right center;
    background-color: #e6ebce;
    right: 0;
  }

  to{
    transform: scaleX(1);
    width: 95px;
    transform-origin: right center;
    background-color: #fe4063;
    right: 0;
  }
}

.fifth_letter03{
  width: 25px;
  height: 95px;
  background-color: #fe4063;
  position: absolute;
  right: 70px;
  bottom: 245px;
  z-index: 67;
}

.fifth_letter04{
  width: 150px;
  height: 25px;
  background-color: #e6ebce;
  position: absolute;
}

```

```

        right: 375px;
        bottom: 245px;
        transform-origin: left center;
        transform: scaleX(0);
        animation: fifth_letter04 2s ease-in-out;
        animation-delay: 3s;
        animation-fill-mode: forwards;
        z-index: 66;
    }

    @keyframes fifth_letter04{
        from{
            transform: scaleX(0);
            width: 150px;
            transform-origin: left center;
        }

        49%{
            transform: scaleX(1);
            width: 150px;
            transform-origin: left center;
        }

        50%{
            transform: scaleX(1);
            width: 150px;
            transform-origin: right center;
            background-color: #e6ebce;
            right: 0;
        }

        to{
            transform: scaleX(1);
            width: 95px;
            transform-origin: right center;
            background-color: #fe4063;
            right: 0;
        }
    }

    .fifth_letter05{
        width: 25px;
        height: 95px;
        background-color: #e6ebce;
        position: absolute;
        right: 0;
        bottom: 245px;
        transform: scaleY(0);
        animation: fifth_letter05 2s ease-in-out;
        animation-delay: 2.5s;
        animation-fill-mode: forwards;
        z-index: 67;
    }

    @keyframes fifth_letter05{
        from{
            transform: scaleY(0);
        }

        50%{
            transform: scaleY(1);
            background-color: #e6ebce;
        }

        to{
            transform: scaleY(1);
            background-color: #fe4063;
        }
    }

    .rect{
        width: 290px;
        height: 255px;
        position: absolute;
        bottom: 55px;
        left: 0;
        right: 0;
        margin: 0 auto;
        filter: drop-shadow(0px 5px 3px rgba(0,0,0,0.3));
        z-index: 40;
    }

    .rect > .line01{
        display: block;
        width: 290px;
        height: 25px;
        background-color: #e6ebce;
    }

```

```

        position: absolute;
        top: 0;
        right: 0;
        transform-origin: right center;
        animation: rect_line01 1.5s ease-in-out;
        animation-delay: 1.5s;
        animation-fill-mode: forwards;
    }

@keyframes rect_line01{
    from{
        right: 0;
        transform: scaleX(1);
    }

    to{
        right: -234px;
        transform: scaleX(0.08);
    }
}

.rect > .line02{
    display: block;
    width: 25px;
    height: 255px;
    background-color: #e6ebce;
    position: absolute;
    top: 0;
    right: 0;
    animation: rect_line02 1.1s ease-in-out;
    animation-delay: 2s;
    animation-fill-mode: forwards;
}

@keyframes rect_line02{
    from{
        height: 255px;
        top: 0;
    }

    99%{
        height: 25px;
        top: -30px;
        opacity: 1;
    }

    to{
        height: 25px;
        top: -30px;
        opacity: 0;
    }
}

```

- **style.css #1-3**

```

.rect > .line03{
    display: block;
    width: 290px;
    height: 25px;
    background-color: #e6ebce;
    position: absolute;
    bottom: 0;
    right: 0;
    transform-origin: right center;
    animation: rect_line03 2s ease-in-out;
    animation-delay: 1s;
    animation-fill-mode: forwards;
}

@keyframes rect_line03{
    from{
        right: 0;
        transform: scaleX(1);
    }

    50%{
        right: -75px;
        transform: scaleX(1);
    }

    to{
        right: -155px;
        transform: scaleX(0);
    }
}

```

```

}

.rect > .line03_1{
    display: block;
    width: 25px;
    height: 150px;
    background-color: #e6ebce;
    position: absolute;
    bottom: 0;
    right: -165px;
    transform-origin: center bottom;
    transform: scaleY(0);
    animation: rect_line03_1 2s ease-in-out;
    animation-delay: 3s;
    animation-fill-mode: forwards;
}

@keyframes rect_line03_1{
    from{
        transform: scaleY(0);
        height: 150px;
        transform-origin: center bottom;
    }

    49%{
        transform: scaleY(1);
        height: 150px;
        transform-origin: center bottom;
    }

    50%{
        transform: scaleY(1);
        height: 150px;
        transform-origin: center top;
        background-color: #e6ebce;
        bottom: 0;
    }

    to{
        transform: scaleY(1);
        height: 95px;
        transform-origin: center top;
        background-color: #fe4063;
        bottom: 190px;
    }
}

.rect > .line04{
    display: block;
    width: 25px;
    height: 255px;
    background-color: #e6ebce;
    position: absolute;
    top: 0;
    left: 0;
    animation: rect_line04 1.1s ease-in-out;
    animation-delay: 2s;
    animation-fill-mode: forwards;
}

@keyframes rect_line04{
    from{
        height: 255px;
        top: 0;
    }

    99%{
        height: 25px;
        top: 40px;
        opacity: 1;
    }

    to{
        height: 25px;
        top: 40px;
        opacity: 0; }
}

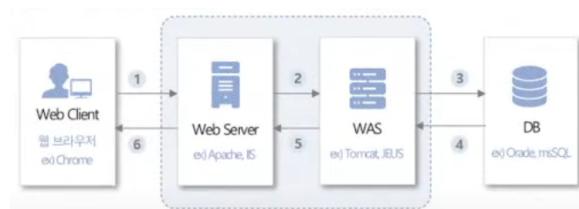
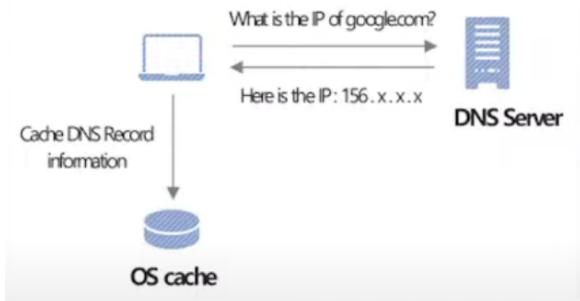
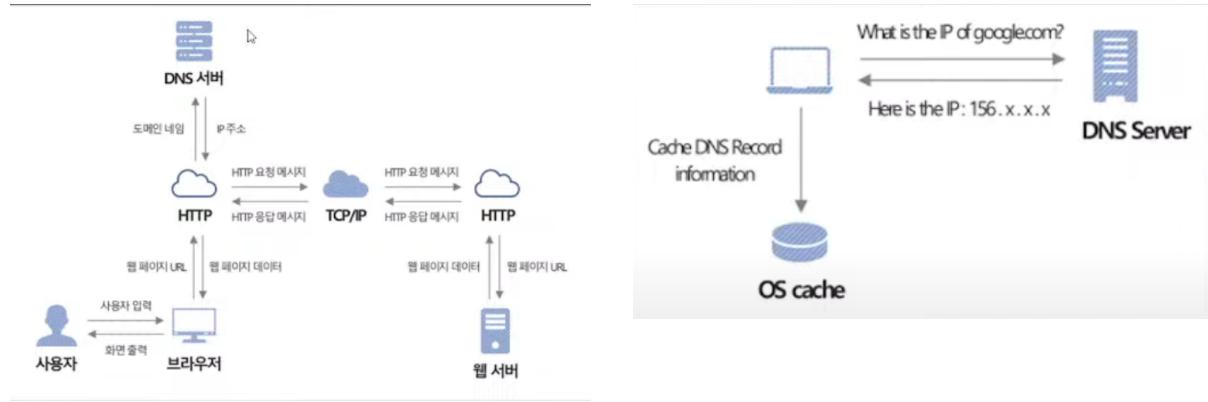
```

• 웹(Web) 브라우저 동작 구조

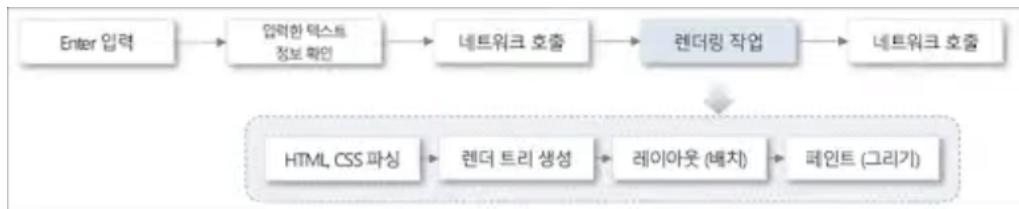
1) 웹 페이지가 연결되는 과정에서의 동작 구조

⇒ 주소창에 웹 브라우저 주소(ex. <https://www.naver.com>) 입력

- ⇒ HTTP 프로토콜을 이용하여 사용자가 입력한 주소를 DNS서버로 보내줌
- ⇒ DNS 서버에서는 요청한 주소와 매칭되는 IP주소를 HTTP 서버로 전달
- ⇒ 전달받은 IP주소를 이용하여 웹 서버에서는 화면에 출력할 html 문서 요청
- ⇒ 웹 서버에서 가장 먼저 받았다는(default) 웹 페이지를 TCP/IP 프로토콜을 통해 서버로 전달
- ⇒ 웹 서버와 WAS 서버간의 연동(페이지 Logic, DB연동 등)과정을 통해 웹 브라우저로 결과 전달
- ⇒ 웹 브라우저에서 전달 받은 값 중에 WAS에서 작업한 결과물을 웹 서버로 전송
- ⇒ 웹 서버는 웹 브라우저에게 html 문서 결과(HTTP Status Codes) 전달
- ⇒ 랜더링과정(성능 최적화, 웹 언어로의 구조 변환 등)을 통해 node를 화면에 배치 → 웹 브라우저 화면 출력



HTTP Status Codes



- DNS 서버 통신 및 동작 구조

