

Hello,

**KDT 웹 개발자 양성 프로젝트**

5기!

with





**페이지  
클론!**

# FOOTER 구현하기



## COMPANY

[한눈에 보기](#)

[스타벅스 사명](#)

[스타벅스 소개](#)

[국내 뉴스룸](#)

[세계의 스타벅스](#)

[글로벌 뉴스룸](#)

## CORPORATE SALES

[단체 및 기업 구매 안내](#)

[단체 주문 배달 안내](#)

## PARTNERSHIP

[신규 입점 제의](#)

[협력 고객사 등록신청](#)

## ONLINE COMMUNITY

[페이스북](#)

[트위터](#)

[유튜브](#)

[인스타그램](#)

## RECRUIT

[채용 소개](#)

[채용 지원하기](#)

STARBUCKS®

[개인정보처리방침](#) | [영상정보처리기기 운영관리 방침](#) | [홈페이지 이용약관](#) | [위치정보 이용약관](#) | [스타벅스 카드 이용약관](#) | [비회원 이용약관](#) | [My DT Pass 서비스 이용약관](#) | [윤리경영 핫라인](#)

[찾아오는 길](#)

[신규입점제의](#)

[사이트 맵](#)

Cloned by **tetz** from Starbucks Korea, © 2022 Starbucks Coffee Company. All Rights Reserved.



JS





# javascript!











희: 희희

망: 망했다

희망이란 밝은 미래로 나아가기 위한 힘입니다.  
성장한다는 의미로 인식될 때마다 뭔가에 실패하거나  
나를 더 나은 사람으로 만드는 계기가 되어주었습니다.



내장  
방식

?

링크  
방식

# 내장 방식!



```
<script>
    alert("Hello, JS World!");
</script>
```

- 위치는 어디서나 사용이 가능합니다!
  - Head 태그 내부
  - Body 태그 내부
  - Head 와 body 사이
  - Body 아래 등

# 링크 방식!



- Java Script 파일을 따로 만들어서 링크하는 방식 like CSS

```
<script src=".main.js"></script>
```

- 위치는 어디서나 사용이 가능합니다!
  - Head 태그 내부
  - Body 태그 내부
  - Head 와 body 사이
  - Body 아래 등

```
<!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>Document</title>
  <script>
    alert("헤드 그런데 js 파일 링크 위")
  </script>
  <script src="./index.js"></script>
  <script>
    alert("헤드 그런데 js 파일 링크 아래")
  </script>
</head>

<script>
  alert("헤드랑 바디 사이");
</script>

<body>
  hello, HTML World!
  <script>
    alert("바디 안쪽");
  </script>
</body>

<script>
  alert("바디 아래");
</script>

</html>
```

# Index.html

# Index.js



```
alert("JS 파일 안쪽!");
```



# JS 기초!



# 표기법

dash-case(kebab-case)

snake\_case

camelCase

ParcelCase



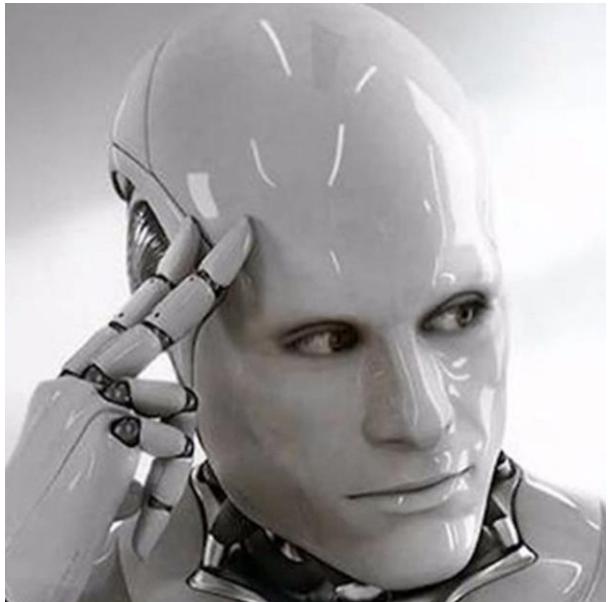
# Zero-based Numbering

0 기반 번호 매기기!

특수한 경우를 제외하고 0부터 숫자를 시작합니다.



1 2 3 4 5 6 7 8 9 10 ~



0 1 2 3 4 5 6 7 8 9 ~



```
// 한 줄 메모
```

```
/* 한 줄 메모 */
```

```
/**
```

```
* 여러 줄
```

```
* 메모1
```

```
* 메모2
```

```
*/
```



# 변수

데이터를 저장하고 참조(사용)하는 데이터의 이름

`var, let, const`



# 예약어

특별한 의미를 가지고 있어, 변수나 함수 이름 등으로 사용할 수 없는 단어  
**Reserved Word**



# 데이터 종류(자료형)

**String**

**Number**

**Boolean**

**Undefined**

**Null**

**Object**

**Array**



# typeof

# 자료형을 알려주는 `typeof`



- 해당 자료형이 어떤 것인지 알려주는 착한 친구입니다!

```
let num = 1;
let str = "이효석";
let bool = true;
let undef = undefined;
let nul = null;
let arr = [1, 2, 3];
let obj = {
  num: 1,
  str: "String"
}

console.log(typeof num);
console.log(typeof str);
console.log(typeof bool);
console.log(typeof undef);
console.log(typeof nul);
console.log(typeof arr);
console.log(typeof obj);
```

```
number
string
boolean
undefined
object
object
object
```



# 문자와

# 변수를 동시에!



# 형 변환

# 성적을 구하는 프로그램 만들기



```
let mathScore = prompt("수학 점수를 입력하세요");
let engScore = prompt("영어 점수를 입력하세요");

let avg = (mathScore + engScore) / 2;

console.log(avg);
```

- 그런데 결과값이 좀 이상하죠?
- **Prompt**로 입력 받은 값은 “**문자**”로 저장이 됩니다!
- “80” + “50” = “8050” → “8050” / 2 → 4025

# 명시적 형변환



- 자동 형변환에 의존 하지 않고 개발자가 직접 형 변환을 시키는 것!
- 문자로 변환 → **String()**;
- 숫자로 변환 → **Number()**;

```
// 문자 데이터로 변환
let num = 123;
num = String(num);
console.log(typeof num);

// 숫자 데이터로 변환
num = Number(num);
console.log(typeof num);

console.log(Number("abcdefg"));
```

```
string
number
NaN
```



# Variable

외 않 되? •





**지금 시작합니다**



# 기본 연산자

# 기본 연산자



- **%** 나머지 연산자

- 홀수 판단 : `num % 2 == 1` 이면 홀수
- 짝수 판단 : `num % 2 == 0` 이면 짝수

- **\*\*** 거듭 제곱

- **\*\*** 를 사용
- $2 ** 3 = 8$
- $3 ** 3 = 27$

# 연산자 줄여서 쓰기



- $\text{num} = \text{num} + 5 \rightarrow \text{num} += 5$
- $\text{num} = \text{num} - 5 \rightarrow \text{num} -= 5$
- $\text{num} = \text{num} * 5 \rightarrow \text{num} *= 5$
- $\text{num} = \text{num} / 5 \rightarrow \text{num} /= 5$

# 증가, 감소 연산자



- Num++;
- Num--;

```
let result1, result2;  
let num = 10, num2 = 10;  
  
result = num++;  
console.log(result);  
  
result2 = ++num2;  
console.log(result2);
```

A dark terminal window with two lines of text in light blue:  
10  
11  
A small blue cursor arrow is visible at the bottom left of the window.



# 비교 연산자

# 비교 연산자!



## 비교연산자

$a == b$  : a와 b가 동일하면 True

$a != b$  : a와 b가 동일하지 않으면 True

$a < b$  : a가 b보다 작으면 ( b가 a보다 크면 ) True

$a <= b$  : a가 b보다 작거나 같으면 True

$a >= b$  : a가 b보다 크거나 같으면 True

## 논리연산자

$a \&& b$  : a, b 전부가 true 일 때만 true

$a || b$  : a, b 둘 중 하나만 true 면 true

# 일치 연산자(==)



- 변수의 값 뿐만 아니라 자료형 까지도 비교!

```
let a = 1;
let b = "1";

// 비교 연산자
console.log(a == b);

// 일치 연산자
console.log(a === b);
```

```
true
false
```



# 논리연산자



# 논리 연산자

|| (OR)

&& (AND)

! (NOT)



# 논리 연산자

|| (OR)

여러개 중 하나라도 true 면 true  
즉, 모든값이 false 일때만 false 를 반환



# 논리 연산자

**&& (AND)**

모든값이 **true** 면 **true**

즉, 하나라도 **false** 면 **false** 를 반환

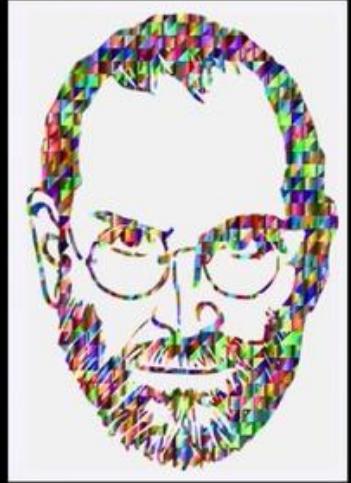


# 논리 연산자

! (NOT)

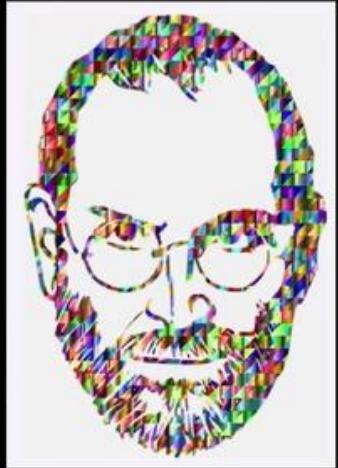
true 면 false

false 면 true



true

스티브 잡스는 **한국인** 이거나**OR**, **남자**이다.



FALSE

스티브 잡스는 **한국인**이고**AND**, **남자**이다.



# 평가

**OR**는 첫번째 **true**를 발견하는 즉시 평가를 멈춤

스티브 잡스는 남자 이거나**OR**, **한국인** 이거나, **군인** 이거나..



# 평가

**AND**는 첫번째 **false**를 발견하는 즉시 평가를 멈춤

스티브 잡스는 남자이고 **AND**, **한국인**이며, **군인**인 동시에..

# 평가

전체 군인의 60%

운전면허가 있고 시력이 좋은 여군

전체 군인의 80%

전체 군인의 7%

-> 여군인데 시력이 좋고 운전면허가 있는 사람



# 조건문

IF!

# IF / ELSE



```
if ( 조건1 ) {  
    // 조건1이 참이라면 실행  
} else {  
    // 조건1이 거짓이라면 실행  
}
```

# IF / ELSE IF / ELSE



```
if ( 조건1 ) {  
    // 조건1이 참이라면 실행  
} else if ( 조건2 ) {  
    // 조건2가 참이라면 실행  
} else {  
    // 조건 1과 2가 모두 참이 아닐 때 실행  
}
```

# IF 중첩



```
if ( 조건1 ) {  
    if ( 조건2 ) {  
        //실행  
    } else {  
        //실행2  
    }  
}
```



```
let isShow = true;
let checked = false;

if (isShow) {
    console.log('Show!'); // Show!
}

if (checked) {
    console.log('Checked!');
}
```



```
let isShow = true;

if (isShow) {
  console.log('Show!');
} else {
  console.log('Hide?');
}
```



# 조건문과 연산자

# OR 연산자, ||



```
// OR 연산자
// 이름이 뽀로로 이거나, 성인이면 통과

let name = "뽀로로";
let age = 18;

if(name === "뽀로로" || age > 19) {
    console.log("통과");
} else {
    console.log("돌아가");
}
```

통과

# AND 연산자, &&



```
// AND 연산자
// 이름이 뽀로로 이고, 성인이면 통과

let name = "뽀로로";
let age = 18;

if(name === "뽀로로" && age > 19) {
    console.log("통과");
} else {
    console.log("돌아가");
}
```

돌아가

# NOT 연산자, !



```
// NOT 연산자
let age = 16;
let isAdult = age > 19;

if(!isAdult) {
    console.log("돌아가");
} else {
    console.log("통과");
}
```

```
돌아가
'
```

# 요상하네요!?



```
// 여성이고, 이름이 Jane 이거나 성인이면 통과
let gender = "M";
let name = "Tom";
let isAdult = true;

if(gender === "F" && name === "Jane" || isAdult === true) {
    console.log("통과");
} else {
    console.log("돌아가");
}
```

통과

```
if( (gender === "F" && name === "Jane") || isAdult === true) {
    console.log("통과");
} else {
    console.log("돌아가");
}
```

# 요상하네요!?



```
// 여성이고, 이름이 Jane 이거나 성인이면 통과
let gender = "M";
let name = "Tom";
let isAdult = true;

if(gender === "F" && (name === "Jane" || isAdult === true) ) {
    console.log("통과");
} else {
    console.log("돌아가");
}
```

돌아가

# 실습!



- 클럽 가드 프로그램을 만들어 봅시다!
- If / else 만 사용해서 구현해야 합니다(else if 는 사용 X)
- 성인(isAdult)이거나, VIP 면 → 통과
- 성인이면서 VIP 여도 취했으면(isDrunken) → 돌아가
- 성인이 아니고 VIP도 아니면서 취했어도, 돈(money)을 줬으면 →  
통과



# 조건문

# Switch



```
switch ( 변수 ) {  
    case 값1:  
        // 변수와 값1이 일치하면 실행  
        break;  
  
    case 값2:  
        // 변수와 값2가 일치하면 실행  
        break;  
  
    default:  
        //일치하는 값이 없을 때 실행  
        break;  
}
```



```
// Switch
let day;
switch (new Date().getDay()) {
  case 0:
    day = "일요일";
    break;
  case 1:
    day = "월요일";
    break;
  case 2:
    day = "화요일";
    break;
  case 3:
    day = "수요일";
    break;
  case 4:
    day = "목요일";
    break;
  case 5:
    day = "금요일";
    break;
  case 6:
    day = "토요일";
    break;
}
console.log(day);
```

A screenshot of a terminal window with a dark background. The text "월요일" is displayed in white, and there is a blue cursor arrow pointing to the left of the text.



# 3항 연산자

# IF 문을 간단하게 표현하는 방법



- 조건식 ? 조건이 참인 경우 실행 : 조건이 거짓인 경우 실행;
- 한 줄로 간단히 표현 가능!

```
// 3항 연산자
let tetzName = "LHS";

if (tetzName == "LHS") {
  console.log("맞았어요 😊");
} else {
  console.log("틀렸어요 😥");
}

tetzName != "LHS" ? console.log("맞았어요 😊") : console.log("틀렸어요 😥");
```

# 실습



- If 문을 이용해서 오늘曜일을 alert 으로 출력하는 프로그램 작성
- 힌트
  - 오늘의 날짜 얻는 방법 : new Date().getDay(); 를 사용하면 오늘의曜일을 숫자로 받을 수 있다
  - 0 : 일요일 / 1 : 월요일 / 2 : 화요일 / 3 : 수요일 / 4 : 목요일 / 5 : 금요일 / 6 : 토요일



# 반복문

# For, while



# For 문



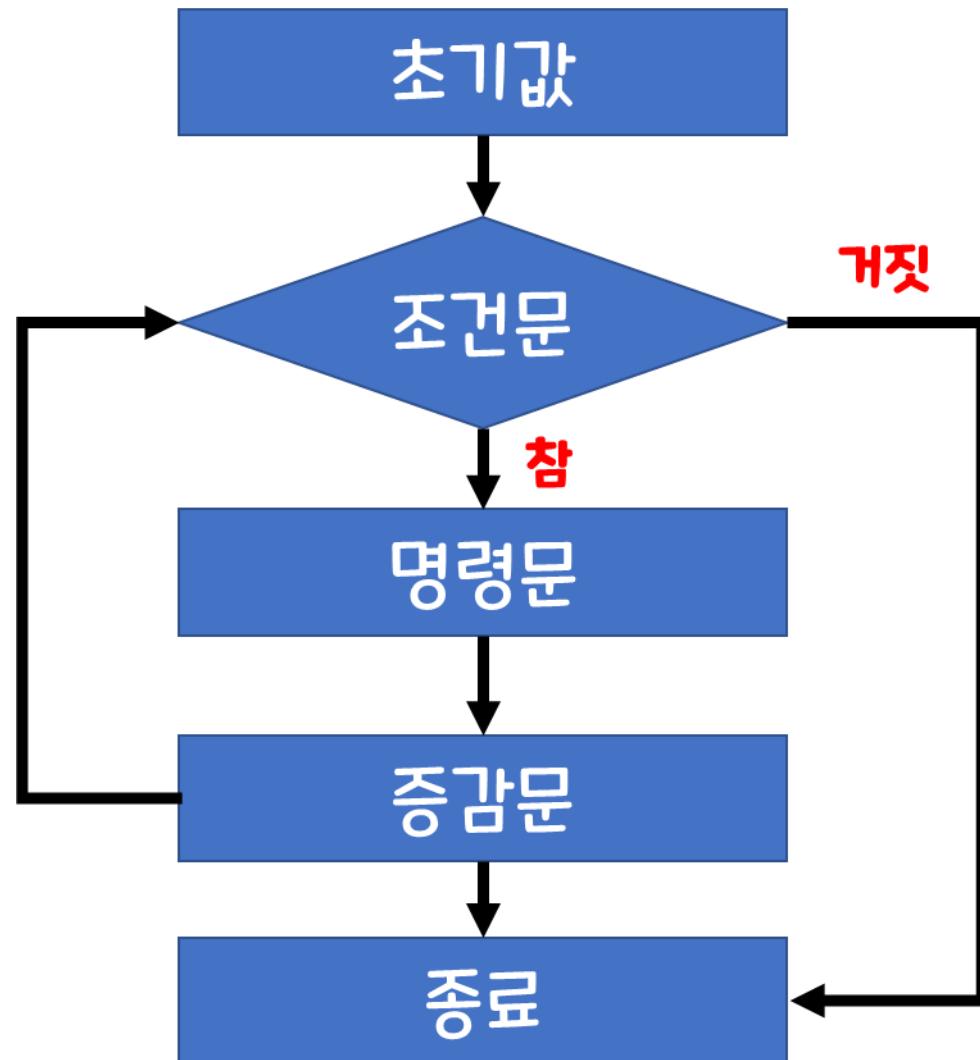
- **for(초기 상태 설정; 조건문; 증감문) {**

**실행할 코드**

}

```
// for 문
for(let index = 0; index < 10; index++) {
    console.log("인사를 ", index+1, "번째 드립니다!
    😊");
}
```

인사를 1 번째 드립니다!	😊
인사를 2 번째 드립니다!	😊
인사를 3 번째 드립니다!	😊
인사를 4 번째 드립니다!	😊
인사를 5 번째 드립니다!	😊
인사를 6 번째 드립니다!	😊
인사를 7 번째 드립니다!	😊
인사를 8 번째 드립니다!	😊
인사를 9 번째 드립니다!	😊
인사를 10 번째 드립니다!	😊



# 2중 For 문



- For 문을 2번 중첩해서 사용하는 반복문!

```
for (let i = 0; i < 3; i++) {  
    console.log(`상위 for 문입니다!', ${i + 1} 번째 실행중`);  
    for (let j = 0; j < 5; j++) {  
        console.log(` 하위 for 문입니다!', ${j + 1} 번째 실행중`);  
    }  
}
```

```
상위 for 문입니다!', 0 번째 실행중  
하위 for 문입니다!', 0 번째 실행중  
하위 for 문입니다!', 1 번째 실행중  
하위 for 문입니다!', 2 번째 실행중  
하위 for 문입니다!', 3 번째 실행중  
하위 for 문입니다!', 4 번째 실행중  
상위 for 문입니다!', 1 번째 실행중  
하위 for 문입니다!', 0 번째 실행중  
하위 for 문입니다!', 1 번째 실행중  
하위 for 문입니다!', 2 번째 실행중  
하위 for 문입니다!', 3 번째 실행중  
하위 for 문입니다!', 2 번째 실행중  
하위 for 문입니다!', 3 번째 실행중  
하위 for 문입니다!', 4 번째 실행중  
하위 for 문입니다!', 5 번째 실행중  
상위 for 문입니다!', 3 번째 실행중  
하위 for 문입니다!', 1 번째 실행중  
하위 for 문입니다!', 2 번째 실행중  
하위 for 문입니다!', 3 번째 실행중  
하위 for 문입니다!', 4 번째 실행중  
하위 for 문입니다!', 5 번째 실행중
```

# 실습



- 구구단을 반복문을 이용해서 console.log 로 출력해 보자!

```
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
```

# 실습



- 0 ~ 100의 숫자 중에서 2 또는 5의 배수 총합 구하기!
- 힌트
  - 나머지 연산자 % 를 사용
  - $5 \% 3 \rightarrow 2$  (5를 3으로 나눈 나머지인 2의 값을 반환)



반복문

while



# while

```
let i = 0;
```

```
while (i < 10) {  
    // 코드
```

```
}
```



# while

```
let i = 0;
```

```
while (i < 10) {  
    // 코드  
    i++;  
}
```

# while 문



- while(조건문) {  
    실행할 코드(명령문)  
}
- For 문과는 달리 조건을 변경하는 구문이 기본적으로 포함이 되어 있지 않기 때문에 무한 루프의 가능성이 농후!
- 주의하여 사용 필요

```
// while 문

// 1번 타입, 조건문을 사용
let index = 0;

while (index < 10) {
    console.log("인사를 ", index + 1, "번째 드립니다! 😊");
    index++;
}

// 2번 타입, 조건문을 사용하지 않고 if 문 + break 사용
let index2 = 0;

while (true) {
    console.log("절을 ", index2 + 1, "번째 드립니다! 😊");
    index2++;
    if (index2 == 10) {
        break;
    }
}
```



인사를	1	번째 드립니다!	😊
인사를	2	번째 드립니다!	😊
인사를	3	번째 드립니다!	😊
인사를	4	번째 드립니다!	😊
인사를	5	번째 드립니다!	😊
인사를	6	번째 드립니다!	😊
인사를	7	번째 드립니다!	😊
인사를	8	번째 드립니다!	😊
인사를	9	번째 드립니다!	😊
인사를	10	번째 드립니다!	😊
절을	1	번째 드립니다!	😊
절을	2	번째 드립니다!	😊
절을	3	번째 드립니다!	😊
절을	4	번째 드립니다!	😊
절을	5	번째 드립니다!	😊
절을	6	번째 드립니다!	😊
절을	7	번째 드립니다!	😊
절을	8	번째 드립니다!	😊
절을	9	번째 드립니다!	😊
절을	10	번째 드립니다!	😊



반복문

do - while



# do.. while

```
let i = 0;
```

```
do {  
    // 코드  
    i++;  
} while (i < 10)
```



# do.. while

```
let i = 0;
```

```
do {  
    // 코드  
    i++;  
} while (i < 10)
```

코드가

최소

1번 실행

# 실습!



- While 을 사용해서 구구단 구현하기!

8 x 1 = 8

8 x 2 = 16

8 x 3 = 24

8 x 4 = 32

8 x 5 = 40

8 x 6 = 48

8 x 7 = 56

8 x 8 = 64

8 x 9 = 72

9 x 1 = 9

9 x 2 = 18

9 x 3 = 27

9 x 4 = 36

9 x 5 = 45

9 x 6 = 54

9 x 7 = 63

9 x 8 = 72

9 x 9 = 81



반복문

제어



# break, continue

**break**

: 멈추고 빠져나옴

**continue**

: 멈추고 다음 반복으로 진행

# break



- 반복문을 멈추고 밖으로 빠져 나감

```
// break

for(let i = 0; i < 100; i++) {
    if(i === 10) {
        console.log("멈춰!");
        break;
    }
    console.log(i);
}
```

```
0
1
2
3
4
5
6
7
8
9
멈춰!
```



# continue

- 반복문을 한 번만 멈추고 다음으로 진행

```
// continue
for (let i = 0; i < 10; i++) {
  if (i % 2 == 0) continue;
  console.log(`#${i} 번 입니다!`);
```

1	번	입니다!
3	번	입니다!
5	번	입니다!
7	번	입니다!
9	번	입니다!
>		

# 실습!



- 1부터 1000까지의 숫자 중에서 짝수의 합을 구하는 프로그램을 만드려 주세요.
- 단, `continue` 를 사용하셔야만 합니다.



함수

Function!



# 함수

특정 동작(기능)을 수행하는 일부 코드의 집합(부분)

**function**



# 함수(function)

함수

함수명

매개변수

```
function sayHello(name){  
    console.log(`Hello, ${name}`);  
}
```

```
sayHello('Mike');
```



```
// 함수 선언  
function helloFunc() {  
    // 실행 코드  
    console.log(1234);  
}
```

```
// 함수 호출  
helloFunc(); // 1234
```



```
function returnFunc() {  
    return 123;  
}  
  
let a = returnFunc();  
  
console.log(a); // 123
```



```
// 기명(이름이 있는) 함수  
// 함수 선언!  
function hello() {  
    console.log('Hello~');  
}
```

```
// 익명(이름이 없는) 함수  
// 함수 표현!  
let world = function () {  
    console.log('World~');  
}
```

```
// 함수 호출!  
hello(); // Hello~  
world(); // World~
```



# 매개변수와 인자



```
function sum(num1, num2) {  
    return num1 + num2;  
}  
  
sum(1, 2);
```

매개 변수

인자

# 매개변수와 인자



- 인자 : 함수를 호출 시에 실제로 전달 되는 값
- 매개 변수 : 함수를 정의할 때, 값을 전달하기 위해 사용하는 변수 → 인자를 받아주는 변수

# 매개 변수가 없는 함수!



- 그냥 실행만 됩니다!

```
// 매개 변수가 없는 함수
function showErr() {
    console.log("에러가 발생 했습니다!");
}

showErr();
```

# 매개 변수가 있는 함수!



- 함수의 출력 내용을 바꾸고 싶다면? 매번 다른 함수를 선언 해서 사용 해야 할까요?
- 매개 변수를 전달하여 함수의 실행 값을 변경 할 수 있습니다!

```
function sayHello(name) {  
    console.log(`Hello, ${name}`);  
}  
  
sayHello("Tetz");  
sayHello("Pororo");  
sayHello("Son");
```

```
Hello, Tetz  
Hello, Pororo  
Hello, Son
```

# 매개 변수가 있는 함수!



- 매개 변수를 활용하여 원하는 데이터를 가공할 수도 있습니다!

```
function sqaure(num) {  
    return num ** 2;  
}  
  
let sqaureNum = sqaure(3);  
console.log(sqaureNum);
```

A small terminal window icon showing the number 9 in white text on a dark background.

# 매개 변수, default



- 매개 변수가 있는 함수인데 매개 변수 전달을 안한다면?

```
function sayHello(name) {  
    console.log(`Hello, ${name}`);  
}  
  
sayHello();
```

Hello, undefined

- 기본 값 설정이 가능합니다!

```
function sayHello(name = "friend") {  
    console.log(`Hello, ${name}`);  
}  
  
sayHello();
```

Hello, friend

# 실습!



- 밑변과 높이를 인자로 전달 했을 때, 삼각형의 넓이를 구하는 함수를 만들어 봅시다!
- 반지름의 길이를 인자로 전달 했을 때, 원의 넓이를 구하는 함수를 만들어 봅시다!
- 피타고라스의 정리를 이용해서 직각 삼각형의 밑변과 높이를 인자로 전달 했을 때, 빗변의 길이를 구하는 함수를 작성해 봅시다!
  - 루트를 씌우는 법 → `Math.sqrt(9) = 3;`



# 함수!

## 선언 방법들



## 함수 선언문 vs 함수 표현식

```
function sayHello(){  
    console.log('Hello');  
}
```

함수 선언문



# 함수 선언문 vs 함수 표현식

```
let sayHello = function(){
    console.log('Hello');
}
```

함수 표현식



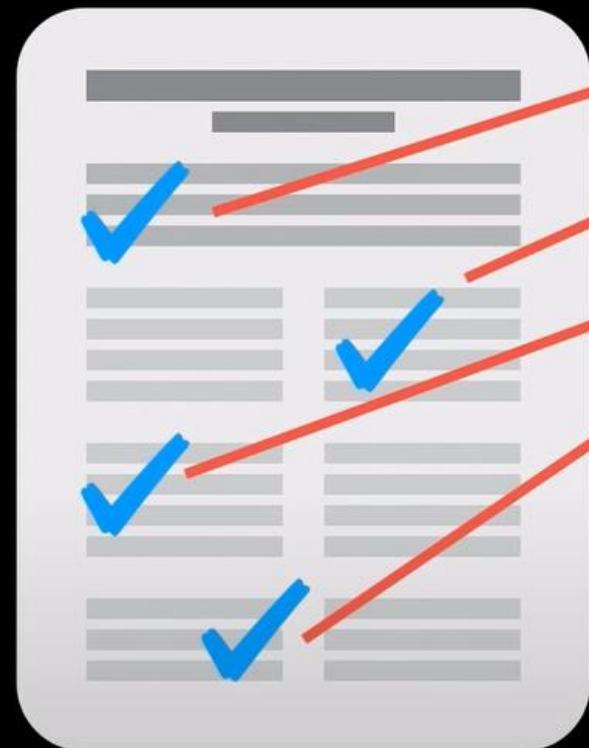
## 함수 선언문 : 어디서든 호출 가능

```
sayHello();  
  
function sayHello(){  
    console.log('Hello');  
}
```





# 함수 선언문 : 어디서든 호출 가능



호이스팅(hoisting)

사용 가능 범위

sayHello();

function sayHello(){  
  console.log('Hello');  
}





## 함수 표현식 : 코드에 도달하면 생성

1 ...

2 ...

3 `let sayHello = function(){` 생성  
 `console.log('Hello');` 사용 가능  
`}`  
4 `sayHello();`



# 화살표 함수(arroW function)

```
let add = function(num1, num2){  
    return num1 + num2;  
}
```



## 화살표 함수(arroew function)

```
let add = (num1, num2) => {  
    return num1 + num2;  
}
```



# 화살표 함수(arroW function)

```
let showError = () => {  
    alert('error!');  
}
```

# 화살표 함수로 만들어 보기



```
// 함수 선언문
function sayHello(name) {
  console.log(`Hello, ${name}`);
}
```

```
// 함수 표현식
let sayHello = function (name) {
  console.log(`Hello, ${name}`);
}
```

```
// 화살표 함수
let sayHello = (name) => {
  console.log(`Hello, ${name}`);
}
```

```
Hello, Tetz
```

# 실습!



- 삼각형의 넓이를 구하는 함수를 함수 표현식으로 변경해 봅시다!
- 원의 넓이를 구하는 함수를 화살표 함수로 변경해 봅시다!
- 피타고라스의 정리를 함수 표현식과 화살표 함수로 변경해 봅시다.



# Onclick!

# onclick



- 각각의 HTML 요소에 속성 값으로 JS 함수를 연결

```
<body>
  <div class="box" onclick="test();">click</div>
</body>
```

```
function test() {
  alert("TEST");
}
```

이 페이지 내용:

TEST

확인

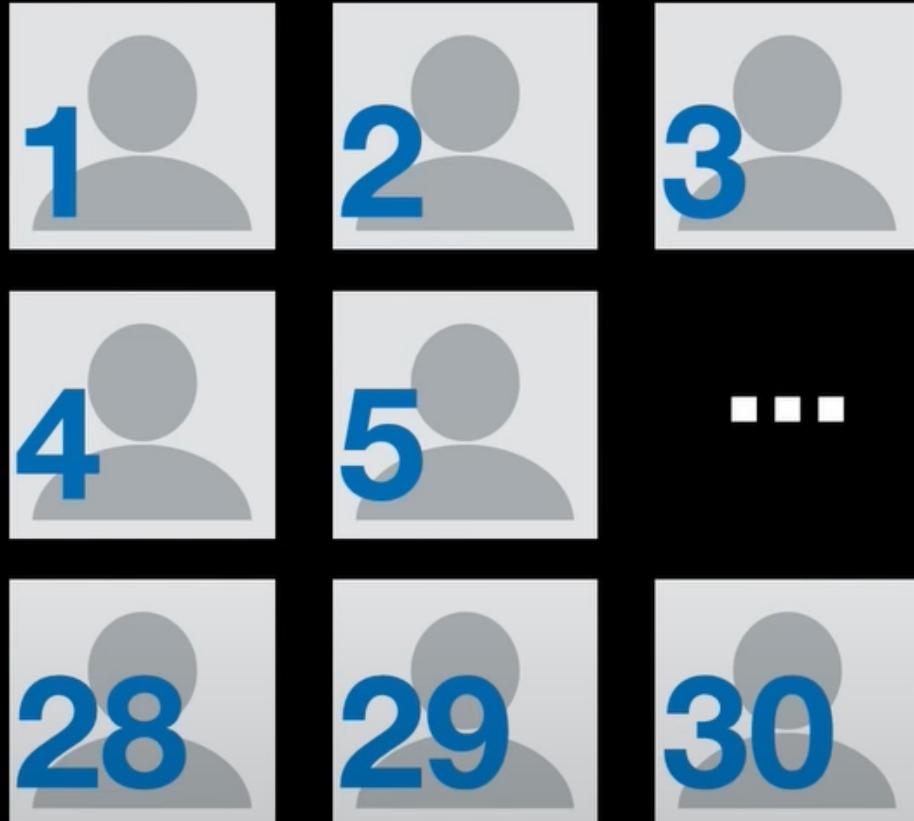


배열

Array



# Array



```
const Mike = "Mike";  
const Jane = "Jane";
```

...

```
const Tom = "Tom";
```



# 배열(array)

: 순서가 있는 리스트



# Array

1번에 철수

2번에 영희

...

30번에 영수

```
let students = ['철수', '영희', ... '영수'];
```

# Array



index

0      1      29

**let students = ['철수', '영희', ... '영수'];**

1번에 철수

2번에 영희

...

30번에 영수

console.log(students[0]); // 철수

console.log(students[1]); // 영희

console.log(students[29]); // 영수

students[0] = '민정';

console.log(students) // ['민정', '영희', ...]

# 배열은 문자 뿐만 아니라, 숫자, 객체, 함수 등도 포함할 수 있음



```
let arr = [
    '민수',
    3,
    false,
    {
        name : 'Mike',
        age : 30,
    },
    function(){
        console.log('TEST');
    }
];
```

# 실습!



- kdt라는 변수에 자신과 같은 줄에 앉은 사람들의 이름이 담긴 배열을 만들기!
- 3번째 참여자의 이름을 출력하기!



배열

사용의 이유!



**length** : 배열의 길이

**students.length // 30**



## push(): 배열 끝에 추가

```
let days = ['월', '화', '수'];
```

```
days.push('목')
```

```
console.log(days) // ['월', '화', '수', '목']
```



## pop(): 배열 끝 요소 제거

```
let days = ['월', '화', '수'];  
days.pop()  
console.log(days) // ['월', '화']
```



## shift, unshift 배열 앞에 제거/추가

추가

```
days.unshift('일');
console.log(days) // ['일', '월', '화', '수'];
```

제거

```
days.shift();
console.log(days) // ['월', '화', '수'];
```

# 실습!



- kdt라는 배열 제일 마지막에 자신을 추가하기!
- kdt라는 배열에서 자신을 빼기!
- kdt 배열 제일 앞에 자신을 추가하기!



## 반복문 : **for**

```
let days = ['월', '화', '수'];  
  
for(let index = 0; index < days.length; index++){  
    console.log(days[index])  
}
```

# 실습!



- 자신과 같은 줄에 앉은 사람의 이름을 출력하는 프로그램 작성하기!
- 출력 내용은
  - X 번째 줄의 n 번째 참여자의 이름은 “~~~”입니다.



# 메소드 체이닝



```
// .split: 문자를 인수 기준으로 쪼개서 배열로 반환
let helloTest = "H-e-l-l-o";
let helloTestArr = helloTest.split("-");
console.log(helloTestArr);
```

```
// .reverse: 배열의 순서를 뒤집어서 반환
// .join: 배열을 인수 기준으로 문자로 병합해서 반환
```

```
let hello = "Hello~";
helloArr = hello.split("");
console.log(helloArr);
```

```
let reverseHello = helloArr.reverse();
console.log(reverseHello);
```

```
let helloJoin = reverseHello.join("");
console.log(helloJoin);
```

```
console.log(hello.split("").reverse().join(""));
```

```
▶ (5) ['H', 'e', 'l', 'l', 'o']
▶ (6) ['H', 'e', 'l', 'l', 'o', '~']
▶ (6) ['~', 'o', 'l', 'l', 'e', 'H']
~olleH
~olleH
```

# 메소드 체이닝



- 각각의 메소드를 연결해서 사용하는 개념! 단, 사용한 메소드가 반환 (return) 값을 가지고 있는 경우에만 사용이 가능!
- `hello.split("")` → `['H', 'e', 'T', 'T', 'o']` 라는 배열이 반환 됨
- 배열에는 `reverse()`라는 메소드가 존재
- `hello.split("").reverse()` 는 `['H', 'e', 'T', 'T', 'o'].reverse()` 와 동일한 것!
- `['H', 'e', 'T', 'T', 'o'].reverse()` → `['o', 'T', 'T', 'e', 'H']` 와 동일
- `hello.split("").reverse().join("")` → `['o', 'T', 'T', 'e', 'H'].join("")` 과 동일

# 메소드 체이닝



- ['H', 'e', 'l', 'l', 'o'].reverse() → ['o', 'l', 'l', 'e', 'H'] 와 동일
- hello.split("").reverse().join("") → ['o', 'l', 'l', 'e', 'H'].join("") 과  
동일

# 실습!



```
const kdtCurriculum = ["HTML", "CSS", "JS", "BACKEND", "REACT"];
```

- 위의 배열에서 첫번째와 마지막 데이터의 문자열을 메소드 체이닝을 이용하여 반대로 뒤집어 주세요.

LMTH  
TCAER

```
const kdtCurriculum = ["HTML", "CSS", "JS", "BACKEND", "REACT"];
const result1 =
const result2 =
console.log(result1);
console.log(result2);
```



# 객체!

# Object



# Object



Superman

name : clark  
age : 33

키(key)

```
const superman = {  
  name: 'clark'  
  age: 33,  
}
```

값(value)



# Object - 접근, 추가, 삭제

```
const superman = {  
    name : 'clark',  
    age : 33,  
}
```

접근

```
superman.name // 'clark'
```

```
superman['age'] // 33
```

추가

```
superman.gender = 'male';
```

```
superman['hairColor'] = 'black';
```

삭제

```
delete superman.hairColor;
```



```
// 객체 생성
let superman = {
    name: "Clark",
    age: 33
}
```

```
// 객체 접근
console.log(superman.name);
console.log(superman.age);
```

```
// 객체 데이터 추가
superman.gender = "M";
superman["height"] = 187;
console.log(superman);
```

```
// 객체 데이터 삭제
delete superman.height;
console.log(superman);
```

```
Clark
33
▶ {name: 'Clark', age: 33, gender: 'M', height: 187}
▶ {name: 'Clark', age: 33, gender: 'M'}
```

# 실습!



- pororo 라는 객체형 변수를 선언
  - name: “뽀로로” / age: 7
- pororo 의 이름과 나이를 출력하기
- pororo 의 성별(gender)을 추가하고 출력하기
- Pororo 의 키(height)를 추가하고 출력하기
- pororo 의 성별 데이터를 삭제하고, pororo 객체를 출력하기



## Object - 프로퍼티 존재 여부 확인

```
const superman = {  
    name : 'clark',  
    age : 33,  
}
```

```
superman.birthDay;  
// undefined
```

```
'birthDay' in superman;  
// false
```

```
'age' in superman;  
// true
```

```
// 객체 생성
let superman = {
    name: "Clark",
    age: 33
}

console.log(superman.birthDay);
console.log('name' in superman);
console.log('height' in superman);
```

```
undefined
true
false
```



# 실습!



- pororo 가 name, height 프로퍼티를 가지고 있는지 출력



# for ... in 반복문

```
for(let key in superman){  
    console.log(key)  
    console.log(superman[key])  
}
```



```
let superman = {  
    name: "Clark",  
    age: 33,  
    height: 187,  
    weight: 77,  
}  
  
for(key in superman) {  
    console.log(key);  
    console.log(superman[key]);  
}
```

name
Clark
age
33
height
187
weight
77

# 실습!



- pororo 의 모든 key 값과 정보를 출력하는 프로그램 만들기



# 객체 Object

# 메소드 Method



# Object



**Superman**

**name : clark**  
**age : 33**

```
const superman = {  
  name : 'clark',  
  age : 33,  
  fly : function( ){  
    console.log('날아갑니다.')  
  }  
}
```



# Object

**method** : 객체 프로퍼티로 할당 된 함수

superman.fly()  
// 날아갑니다.

```
const superman = {  
    name : 'clark',  
    age : 33,  
    fly : function( ){  
        console.log('날아갑니다.')  
    }  
}
```



# Object

```
const superman = {  
    name : 'clark',  
    age : 33,  
    fly : function( ){  
        console.log('날아갑니다.')  
    }  
}
```



# Object

```
const superman = {  
    name : 'clark',  
    age : 33,  
    fly( ){  
        console.log('날아갑니다.')  
    }  
}
```

# 실습!



- pororo 객체에 “뽀로로는 귀엽습니다!” 를 출력하는 method 추가하기



this



# Object

```
const user = {  
    name : 'Mike',  
    sayHello : function(){  
        console.log(`Hello, I'm ${user.name}`);  
    }  
}
```

# Object



```
const user = {  
    name : 'Mike',  
    sayHello : function(){  
        console.log(`Hello, I'm ${this.name}`);  
    }  
}  
  
user.sayHello( );  
// Hello, I'm Mike
```



```
let boy = {  
    name: "Mike",  
    sayHello  
}  
  
let girl = {  
    name: "Jane",  
    sayHello  
  
}  
  
function sayHello() {  
    console.log(`Hello, I'm ${this.name}`);  
}  
  
boy.sayHello();  
girl.sayHello();
```

```
Hello, I'm Mike  
Hello, I'm Jane
```

# 실습!



- pororo 객체에 뽀로로의 이름을 출력하는 method 추가하기
- 특정 객체의 height 를 출력하는 showHeight() 함수를 작성하고  
pororo 객체의 메소드로 추가 → pororo 의 키를 출력하기



# 화살표 함수와

this



```
let sayHello = () => {
  console.log(`Hello, I'm ${this.name}`);
  console.log(this);
}

let boy = {
  name: "Mike",
  sayHello
}

let girl = {
  name: "Jane",
  sayHello
}

boy.sayHello();
girl.sayHello();
```

```
Hello, I'm dom.js:242
Window {window: Window, self: Window, document: document, name: '', location: Location}
Hello, I'm dom.js:242
Window {window: Window, self: Window, document: document, name: '', location: Location}
```



# Object

화살표 함수는 일반 함수와는 달리 자신만의 **this**를 가지지 않음

화살표 함수 내부에서 **this**를 사용하면, 그 **this**는 외부에서 값을 가져 옴



# Object

```
let boy = {  
    name : 'Mike',  
    sayHello : () => {  
        console.log(this); // 전역객체  
    }  
}
```

**boy.sayHello( );**

**this != boy**

- 브라우저 환경 : **window**
- Node.js : **global**





# 생성자

# 함수



# Object



Superman

name : clark  
age : 33

키(key)

```
const superman = {  
  name: 'clark'  
  age: 33,  
}
```

값(value)





# 생성자 함수

첫 글자는 대문자로

```
function User(name, age){  
    this.name = name;  
    this.age = age;  
}
```

```
let user1 = new User('Mike', 30);  
let user2 = new User('Jane', 22);  
let user3 = new User('Tom', 17);
```

new 연산자를 사용해서 호출



# 생성자 함수

```
function User(name, age){  
    this.name = name;  
    this.age = age;  
}
```

```
let user1 = new User('Mike', 30);  
let user2 = new User('Jane', 22);  
let user3 = new User('Tom', 17);
```



# 생성자 함수



```
function User(name, age){  
    this.name = name;  
    this.age = age;  
    this.sayName = function(){  
        console.log(this.name);  
    }  
}  
  
let user5 = new User('Han', 40);  
user5.sayName(); // 'Han'
```



```
function Fruits(name, price) {  
    this.name = name;  
    this.price = price;  
    this.showPrice = function () {  
        console.log(` ${this.name} 의 가격은 ${this.price} 입니다! `);  
    };  
}  
  
let apple = new Fruits("apple", 1000);  
let pineapple = new Fruits("pineapple", 2000);  
let watermelon = new Fruits("watermelon", 10000);  
let strawberry = new Fruits("strawberry", 50000);  
  
apple.showPrice();  
pineapple.showPrice();  
watermelon.showPrice();  
strawberry.showPrice();
```

apple 의 가격은 1000 입니다!

pineapple 의 가격은 2000 입니다!

watermelon 의 가격은 10000 입니다!

strawberry 의 가격은 50000 입니다!

# 실습!



- Kdt라는 생성자 함수를 만들어 주세요.
- 생성자 함수에는
  - 참여자의 이름 / 성별 데이터가 포함 됩니다.
  - 참여자의 이름과 성별을 출력하는 method도 포함됩니다.
- 자신과 짹꿍, 그리고 바로 같은 줄에 있는 참여자 4명에 대한 정보를 생성자 함수를 이용하여 5개의 변수로 저장하세요.
- 참여자의 이름과 성별을 출력하는 method를 이용 5명의 정보를 출력해 보세요.



# DOM

## (Document Object Model)

# DOM(Document Object Model)

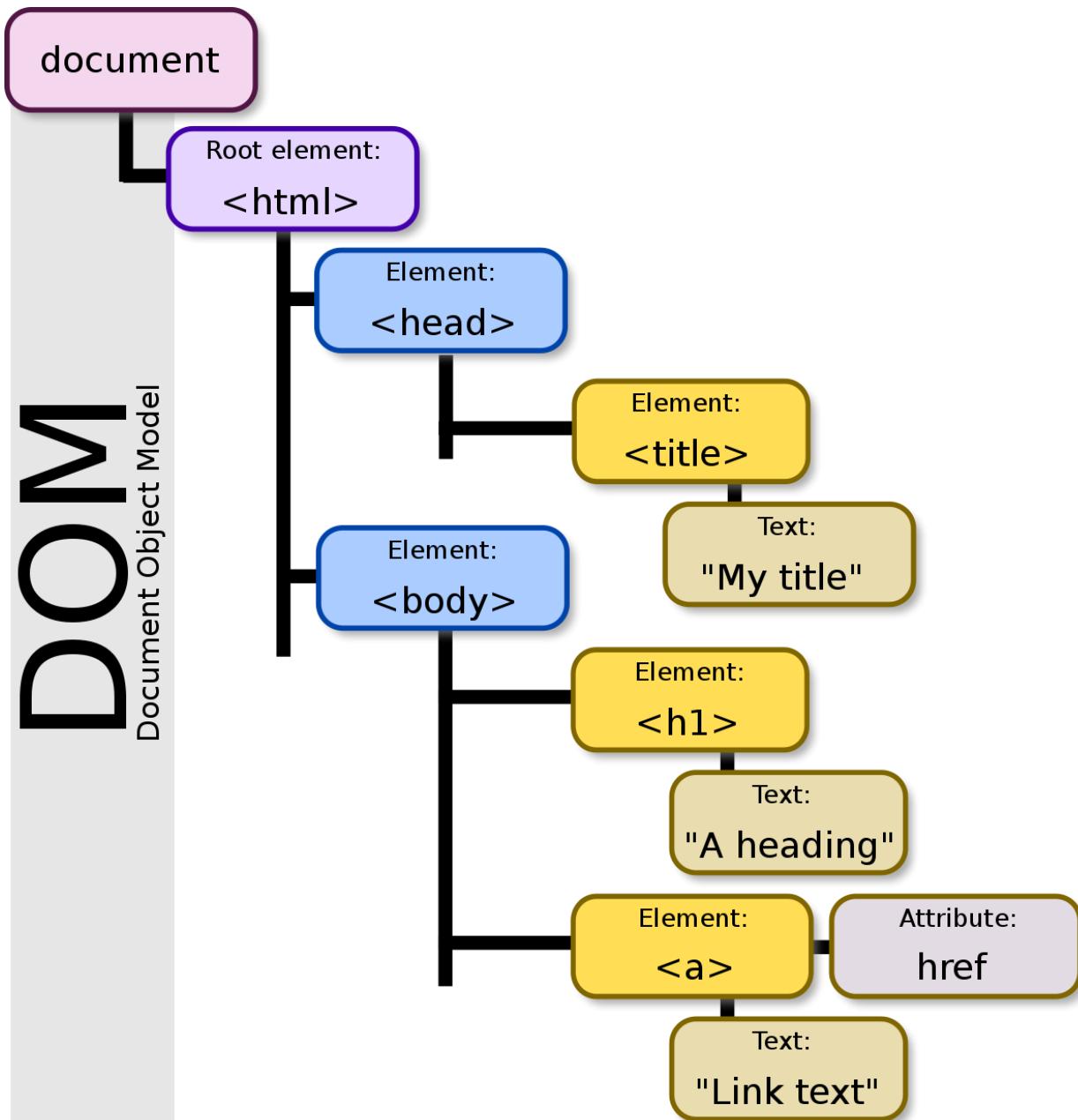


- HTML 문서 요소의 집합!
- HTML 문서는 각각의 node 와 object 의 집합으로 문서를 표현
- 따라서 각각 node 또는 object 에 접근하여 문서 구조 / 스타일 / 내용 등을 변경 할 수 있도록 하는 것!



# DOM

Document Object Model



# 실습



- .box 를 최초 클릭하면 배경을 orange 색으로 변경하기
- .box 를 다시 클릭 했을 때 배경이 orange 색이면 skyblue 로 변경하거나 skyblue 면 orange 로 변경하는 페이지 만들기!



# DOM API

**Document Object Model, Application Programming Interface**



```
// HTML 요소(Element) 1개 검색/찾기
const boxEl = document.querySelector('.box');

// HTML 요소에 적용할 수 있는 메소드!
boxEl.addEventListener();

// 인수(Arguments)를 추가 가능!
boxEl.addEventListener(1, 2);

// 1 - 이벤트(Event, 상황)
boxEl.addEventListener('click', 2);

// 2 - 핸들러(Handler, 실행할 함수)
boxEl.addEventListener('click', function () {
  console.log('Click~!');
});
```



# querySelector

# querySelector("요소 선택자")



- 요소 선택자를 사용해서 자신이 가져오고 싶어하는 요소를 가져오는 메소드
- 문서에서 만나는 **제일 첫번째 요소**를 반환 합니다!

```
let boxEl = document.querySelector(".box");

console.log(boxEl);
```



# getElementById

# getElementById("ID")



- ID 이름을 불러서 해당 ID를 가지는 요소를 불러오는 메소드

```
const inputEl = document.getElementById("input");
```

A screenshot of a browser's developer tools console. The `document` object is expanded, showing several methods: `getElementById` (highlighted with a blue background), `getElementsByName`, `getElementsByClassName`, `getElementsByTagName`, and `getElementsByTagNameNS`. Below these, there are other methods like `createEvent`, `createEventObject`, and `createEventSource`.



# classList

# classList.add / remove / contain



- 선택 요소에 class 를 더하거나, 빼거나, 클래스가 존재하는지 체크 하는 메소드
- 해당 기능과 CSS 를 잘 활용하면 변화무쌍(?)한 웹페이지 구성이 가능



```
// HTML 요소(Element) 검색/찾기  
const boxEl = document.querySelector('.box');  
  
// 요소의 클래스 정보 객체 활용!  
boxEl.classList.add('active');  
let isContains = boxEl.classList.contains('active');  
console.log(isContains); // true  
  
boxEl.classList.remove('active');  
isContains = boxEl.classList.contains('active');  
console.log(isContains); // false
```



# AddEventListener

# addEventListener(이벤트, 명령)



- 선택 요소에 지정한 이벤트가 발생하면, 약속 된 명령어를 실행시키는 메소드

```
let boxEl = document.querySelector(".box");

console.log(boxEl);

boxEl.addEventListener("click", function() {
    alert("click!");
})
```

```
document.querySelector(".box").addEventListener("click", function() {
    alert("click!");
})
```



```
let boxEl = document.querySelector(".box");
console.log(boxEl);
console.log(boxEl.classList.contains("orange"));

boxEl.addEventListener("click", function() {
  boxEl.classList.add("orange");
  console.log(boxEl);
  console.log(boxEl.classList.contains("orange"));
})
```

```
<div class="box">Box!!</div>
false

<div class="box orange">Box!!</div>
true
.
```

# **addEventListener** 이벤트의 종류



- Click : 클릭
- Mouse 계열
  - Mouseover : 요소에 커서를 올렸을 때
  - Mouseout : 마우스가 요소를 벗어날 때
  - Mousedown : 마우스 버튼을 누르고 있는 상태
  - Mouseup : 마우스 버튼을 떼는 순간
- Focus : 포커스가 갔을 때
- Blur : 포커스가 벗어나는 순간

# **addEventListener** 이벤트의 종류



- **Key** 계열
  - Keypress : 키를 누르는 순간 + 누르고 있는 동안 계속 발생
  - Keydown : 키를 누르는 순간에만 발생
  - Keyup : 키를 눌렀다가 빼는 순간
- **Load** : 웹페이지에 필요한 모든 파일(html, css, js 등)의 다운로드가 완료 되었을 때
- **Resize** : 브라우저 창의 크기가 변경 될 때
- **Scroll** : 스크롤이 발생할 때
- **Unload** : 링크를 타고 이동하거나, 브라우저를 닫을 때
- **Change** : 폼 필드의 상태가 변경 되었을 때



# setAttribute

# setAttribute, html 요소 속성 추가



- 선택한 요소의 속성 값을 직접 지정할 수 있는 메소드
- 요소.setAttribute("속성명", "지정할 속성")

```
searchInputEl.setAttribute("placeholder", "통합검색");
```

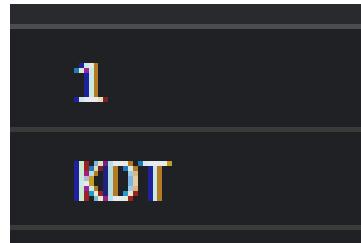


# textContent



```
let boxEl = document.querySelector(".box");
console.log(boxEl.textContent);

boxEl.textContent = "KDT";
console.log(boxEl.textContent);
```



# 실습



- .box 를 최초 클릭하면 배경을 orange 색으로 변경하기
- .box 를 다시 클릭 했을 때 배경이 orange 색이면 skyblue 로 변경 하거나 skyblue 면 orange 로 변경하는 페이지 만들기!
- text 인풋 / 임의의 span 요소 “가나다라마바사” / 버튼 2개 만들기
- 첫번째, 버튼을 클릭하면
  - text 인풋에 placeholder 속성 “아이디를 입력하세요” 추가
  - span 요소의 텍스트도 “아이디를 입력하세요”로 변경하기
- 두번째 버튼을 클릭하면
  - Text 인풋에 입력 받은 값을 span 요소의 컨텐츠로 만들기



# querySelectorAll

# querySelectorAll("요소 선택자")



- 문서에 존재하는 모든 요소를 찾아주는 메소드
- 모든 요소를 가져와서 배열(같은) 데이터로 만들어 줍니다!

```
<body>
  <div class="box">1</div>
  <div class="box">2</div>
  <div class="box">3</div>
  <div class="box">4</div>
  <div class="box">5</div>
  <div class="box">6</div>
  <div class="box">7</div>
</body>
```

```
let boxEls = document.querySelectorAll(".box");
console.log(boxEls);
```

```
▼ NodeList(7) [div.box, div.box, div.box, div.box, div.box, div.box, div.box]
  ▷ 0: div.box
  ▷ 1: div.box
  ▷ 2: div.box
  ▷ 3: div.box
  ▷ 4: div.box
  ▷ 5: div.box
  ▷ 6: div.box
  length: 7
  ▷ [[Prototype]]: NodeList
```

# forEach(function (반복중인 요소, 인덱스) {})



- 찾은 요소 전부에게 명령을 반복적으로 실행해 주는 메소드

```
let boxEls = document.querySelectorAll(".box");
console.log(boxEls);

boxEls.forEach(function (boxEl, index) {
  boxEl.classList.add(`box_${index + 1}`);
})

console.log(boxEls);
```

```
▼ NodeList(7) [div.box, div.box,
  ▶ 0: div.box.box_1
  ▶ 1: div.box.box_2
  ▶ 2: div.box.box_3
  ▶ 3: div.box.box_4
  ▶ 4: div.box.box_5
  ▶ 5: div.box.box_6
  ▶ 6: div.box.box_7
  length: 7
  ▶ [[Prototype]]: NodeList
```

```
▼ NodeList(7) [div.box.box_1, di
, div.box.box_6, div.box.box_7
  ▶ 0: div.box.box_1
  ▶ 1: div.box.box_2
  ▶ 2: div.box.box_3
  ▶ 3: div.box.box_4
  ▶ 4: div.box.box_5
  ▶ 5: div.box.box_6
  ▶ 6: div.box.box_7
  length: 7
  ▶ [[Prototype]]: NodeList
```



```
// HTML 요소(Element) 모두 검색/찾기
const boxEls = document.querySelectorAll('.box');
console.log(boxEls);

// 찾은 요소들 반복해서 함수 실행!
// 익명 함수를 인수로 추가!
boxEls.forEach(function () {});

// 첫 번째 매개변수(boxEl): 반복 중인 요소.
// 두 번째 매개변수(index): 반복 중인 번호
boxEls.forEach(function (boxEl, index) {});

// 출력!
boxEls.forEach(function (boxEl, index) {
  boxEl.classList.add(`order-${index + 1}`);
  console.log(index, boxEl);
});
```

# 실습



- 최 상단에 버튼 하나 만들기
- 7 개의 box 클래스를 가지는 <div> 요소 만들기
- 버튼을 클릭하면 각각의 박스의 배경색이 변경 되는 페이지 만들기
- 첫번째 박스, 두번째 박스, 세번째 박스 ~ 일곱번째 박스 색상을 무  
지개에 맞춰서 변경





```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Document</title>
  <script src="./dom.js"></script>
</head>

<body>
  <div class="box">Box! !</div>
</body>

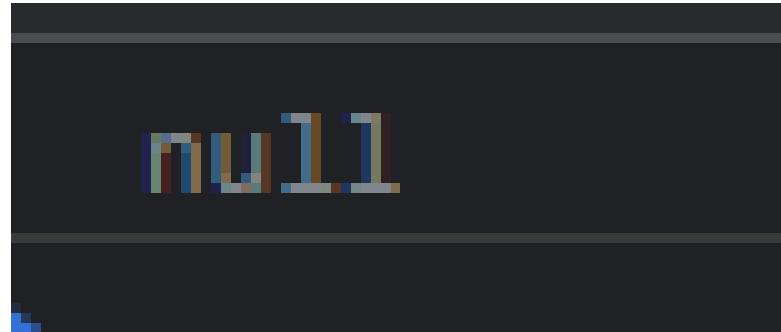
</html>
```

## dom.html

```
let boxEl = document.querySelector(".box");

console.log(boxEl);
```

## dom.js





# dom.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Document</title>
</head>

<body>
  <div class="box">Box!!</div>
  <script src="./dom.js"></script>
</body>

</html>
```

```
<div class="box">Box!!</div>
```

```
let boxEl = document.querySelector(".box");
```

```
console.log(boxEl);
```

# dom.js



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Document</title>
  <script defer src="./dom.js"></script>
</head>

<body>
  <div class="box">Box!!</div>
</body>

</html>
```

## dom.html

```
<div class="box">Box!!</div>
```

```
let boxEl = document.querySelector(".box");

console.log(boxEl);
```

## dom.js





```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
  <script src=".main.js"></script>
</head>
<body>
  <div class="box">Box!!</div>
</body>
</html>
```



```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
</head>
<body>
  <div class="box">Box!!</div>
  <script src=".main.js"></script>
</body>
</html>
```



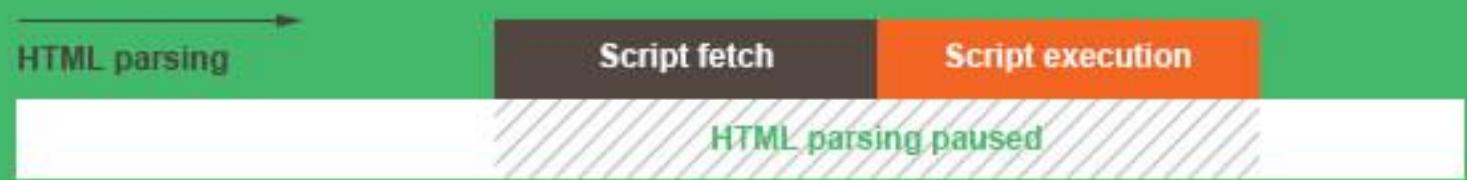
```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
  <script defer src=".main.js"></script>
</head>
<body>
  <div class="box">Box!!</div>
</body>
</html>
```



Defer,  
Async



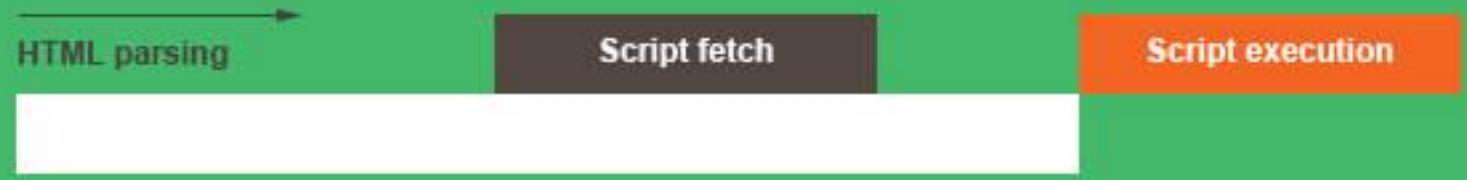
`<script>`



`<script async>`



`<script defer>`





미래의 그날이 온다!

미래  
트리니티  
OCT 21 2015



Prain

한국 주연 배우 • 韩国 主演男演员 • 中国演员 • 中国导演  
A Universal Picture © 1985 & 2015 Universal Studios.







JS





# JS 파일

# 연결하기

# Main.js 파일 연결하기!



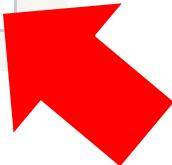
- 안전하게 속성 값으로 defer 넣고 시작!



# 서브메뉴

# Search

[Sign In](#) | [My Starbucks](#) | [Customer Service & Ideas](#) | [Find a Store](#)



# 서브 메뉴, search 변경하기



- 기존은 CSS 의 :focus 를 사용하여 구현
- 따라서, 돌보기를 피해서 input 을 클릭 했을 때에만 작동 → 불편
- 돌보기를 클릭하면 기능이 동작하도록 변경
- 돌보기를 클릭하면 focus 가 가도록 설정하여 기존의 CSS 사용
- 포커스 시 → “통합 검색” 이라는 placeholder 도 추가!(by JS)



공지 사항



공지사항 스타벅스 e-Gift Card 구매 가능 금액안내



# 공지 사항, 슬라이드 메뉴 적용하기



- 해당 기능을 전부 구현하는 것은 매우 어렵고 귀찮은 작업이죠?
- 이럴 때 쓰는 것이 바로! Library
- Swiper라는 라이브러리를 사용해 봅시다!



# Swiper

The Most Modern Mobile  
Touch Slider

[Get Started](#) [API](#) [Demos](#) [React](#) [Solid](#) [Svelte](#) [Vue](#) [Angular](#)

MIT Licensed, v8.3.1 released on July 13, 2022 [Changelog](#)

31,724 stars

## Top Notch Features

- ✓ Library Agnostic
- ✓ Flexbox Layout
- ✓ Multi Row Slides Layout
- ✓ Two-way Control
- ✓ Rich API
- ✓ Parallax Transitions
- ✓ Virtual Slides
- ✓ Mutation Observer
- ✓ Full True RTL Support
- ✓ 3D Effects
- ✓ Full Navigation Control
- ✓ Most Flexible Slides Layout Grid
- ✓ Images Lazy Loading
- ✓ And many more



<https://swiperjs.com/>



## Use Swiper from CDN

If you don't want to include Swiper files in your project, you may use it from CDN. The following files are available:

```
<link  
  rel="stylesheet"  
  href="https://unpkg.com/swiper@8/swiper-bundle.min.css"  
/>  
  
<script src="https://unpkg.com/swiper@8/swiper-bundle.min.js"></script>
```



If you use ES modules in browser, there is a CDN version for that too:

```
<script type="module">  
  import Swiper from 'https://unpkg.com/swiper@8/swiper-bundle.esm.browser.min.js'  
  
  const swiper = new Swiper(...)  
</script>
```

## Download assets

If you want to use Swiper assets locally, you can directly download them from <https://unpkg.com/swiper@8/>



## Add Swiper HTML Layout

Now, we need to add basic Swiper layout to our app:

```
<!-- Slider main container -->
<div class="swiper">
  <!-- Additional required wrapper -->
  <div class="swiper-wrapper">
    <!-- Slides -->
    <div class="swiper-slide">Slide 1</div>
    <div class="swiper-slide">Slide 2</div>
    <div class="swiper-slide">Slide 3</div>
    ...
  </div>
  <!-- If we need pagination -->
  <div class="swiper-pagination"></div>

  <!-- If we need navigation buttons -->
  <div class="swiper-button-prev"></div>
  <div class="swiper-button-next"></div>

  <!-- If we need scrollbar -->
  <div class="swiper-scrollbar"></div>
</div>
```



## Swiper CSS Styles/Size

In addition to [Swiper's CSS styles](#), we may need to add some custom styles to set Swiper size:

```
.swiper {  
    width: 600px;  
    height: 300px;  
}
```



## Initialize Swiper

Finally, we need to initialize Swiper in JS:

```
const swiper = new Swiper('.swiper', {
    // Optional parameters
    direction: 'vertical',
    loop: true,

    // If we need pagination
    pagination: {
        el: '.swiper-pagination',
    },

    // Navigation arrows
    navigation: {
        nextEl: '.swiper-button-next',
        prevEl: '.swiper-button-prev',
    },

    // And if we need scrollbar
    scrollbar: {
        el: '.swiper-scrollbar',
    },
});
```

# Swiper 라이브러리 추가



```
<!-- Icons -->
<link rel="stylesheet"
      href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,
wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />

<!-- SWIPER      -->
<link rel="stylesheet" href="https://unpkg.com/swiper@8/swiper-bundle.min.css" />
<script src="https://unpkg.com/swiper@8/swiper-bundle.min.js"></script>

<!-- main -->
<link rel="stylesheet" href=".css/main.css">
<script defer src=".js/main.js"></script>
```

# HTML 코드 추가!



- Swiper 클래스 div 추가
- Swiper-wrapper div 자식으로 swiper-slide div 추가



```
<div class="inner">
    <div class="inner_left">
        <h1>공지사항</h1>
        <div class="swiper">
            <div class="swiper-wrapper">
                <div class="swiper-slide">My DT Pass 이용 안내</div>
                <div class="swiper-slide">My DT Pass 관련 서비스 점검 안내</div>
                <div class="swiper-slide">시스템 개선 및 점검 안내</div>
                <div class="swiper-slide">전자 영수증 서비스 서비스 점검 안내</div>
                <div class="swiper-slide">스타벅스 e-Gift Card 구매 가능 금액 안내</div>
            </div>
        </div>
        <a href="#"><span class="material-symbols-outlined">add_circle</span></a>
    </div>
    <div class="inner_right">
        <h1>스타벅스 프로모션</h1>
        <a href="#"><span class="material-symbols-outlined">expand_circle_down</span></a>
    </div>
</div>
```

# JS에 Swiper 생성자 함수 추가



- 변수 하나를 만들고 Swiper 생성자 함수 추가
- 설정 값 입력
  - Direction: "vertical"
  - Autoplay: true
  - Loop: true

# CSS 수정



- Swiper 클래스의 CSS 수정
- 크기 값이 필요 → 높이 값 주기!
- 아이템 중앙 정렬!
  - 한 줄 텍스트는??





# 프로모션

# 슬라이드



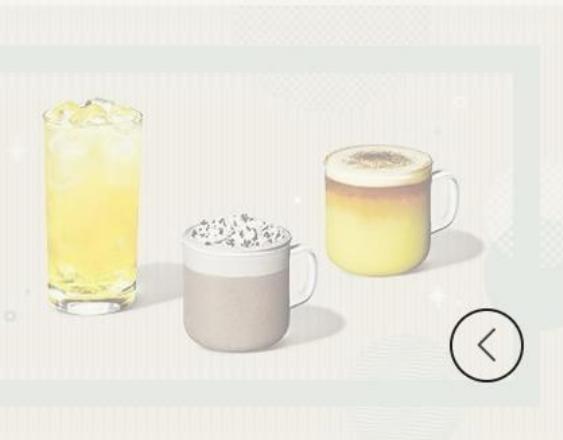
공지사항 탄소중립실천포인트제 도입 안내



스타벅스 프로모션



UR



자세히 보기



회원 계정에 등록된 스타벅스 카드  
1만원당 별★1개를 즉시 추가로

기간 : 2023년 1월 1일 ~ 2023년 12월 31일

자세히 보기

2023년, 스타벅스에서  
가득한 신년을 맞이하세요!

: 2023년 1월 1일(일) ~ 2월 14일(화)

자세히 보기

## STARBUCKS RESERVE™ SPRING

스타벅스 리저브 스프링 원두 & 커피

기간 : 2023년 2월 8일 ~ 2023년 4월 25일

자세히 보기



# HTML 구성하기



- Promotion 클래스 div 선언
- Swiper 사용을 위한 swiper 클래스 div 추가
- Swiper-wapper 추가
- Swiper-slide 추가

# CSS 선언해 놓기!



- .notice .promotion {}
- .notice .promotion .swiper {}
- .notice .promotion .swiper .swiper-wrapper {}
- .notice .promotion .swiper .swiper-wrapper .swiper-slide {}

# Promotion / Swiper 크기 세팅



- 스타벅스 페이지를 참고
- Promotion
  - Height : 658px / background-color: #f6f5ef
- Swiper
  - Height : 553px / Width: calc(819px \* 3 + 20px)

# 중앙 정렬!



- **Text-align: center;** 는 항상 중앙을 가르킬까요?
- 화면 확대 축소에 따른 중앙 정렬 확인!
- Container 의 크기에 따른 중앙 정렬 → left: 50% / translate(-50%, 0);

# 이미지 삽입하기!



- Swiper slide 에 이미지 삽입하기
- 자세히 보기 버튼(black) 추가!



```
<div class="promotion">
  <div class="swiper">
    <div class="swiper-wrapper">
      <div class="swiper-slide">
        
        <a href="#" class="btn btn--black">자세히 보기</a>
      </div>
      <div class="swiper-slide">
        
        <a href="#" class="btn btn--black">자세히 보기</a>
      </div>
      <div class="swiper-slide">
        
        <a href="#" class="btn btn--black">자세히 보기</a>
      </div>
    </div>
  </div>
</div>
```

# Swiper 생성자 함수 추가!



- 새로운 swiperPromotion 변수에 Swiper 생성자 추가
- Direction / slidePerView / spaceBetween / centeredSlide / loop / autoplay 옵션 추가하기!

```
// SWIPE PROMOTION
const swiperPromotion = new Swiper(".promotion .swiper", {
  direction: "horizontal", // 기본 값
  slidesPerView: 3, // 한번에 보여줄
  spaceBetween: 10, // 아이템간 거리
  centeredSlides: true, // 슬라이드 센터 여부
  loop: true, // 루프 여부
  autolay: {
    // 자동 재생, 변경 시간 설정
    delay: 5000,
  }
});
```



# CSS 수정하기!



- Swiper 배경색 제거
- Swiper 동작을 확인하기
- 보여지는 슬라이드의 클래스명 확인(active)
- 양 옆 슬라이드 반투명 처리
- 자세히 보기 버튼 위치 및 크기 조절

# 페이지네이션 / 이동 버튼 추가하기



- Swiper에서 제공하는 pagination / prev / next 버튼 추가하기

```
<div class="swiper-pagination"></div>
<div class="swiper-button-prev"></div>
<div class="swiper-button-next"></div>
```

# 페이지네이션 / 이동 버튼 추가하기



- Swiper에서 제공하는 pagination / prev / next 버튼 추가하기

```
<div class="swiper-pagination"></div>
<div class="swiper-button-prev"></div>
<div class="swiper-button-next"></div>
```

- Swiper 생성자에 옵션 추가하기!

```
// SWIPE PROMOTION
const swiperPromotion = new Swiper(".promotion .swiper", {
  direction: "horizontal", // 기본 값
  slidesPerView: 3, // 한번에 보여줄
  spaceBetween: 10, // 아이템간 거리
  centeredSlides: true, // 슬라이드 센터 여부
  loop: true, // 루프 여부
  autolay: {
    // 자동 재생, 변경 시간 설정
    delay: 5000,
    disableOnInteraction: false,
  },
  pagination: {
    el: ".promotion .swiper-pagination", // pagination을 할 엘리먼트 클래스 설정
    clickable: true, // 클릭 가능 여부 설정
  },
  navigation: {
    prevEl: ".promotion .swiper-button-prev", // 이전 버튼 클래스 설정
    nextEl: ".promotion .swiper-button-next", // 이후 버튼 클래스 설정
  },
});
```



# 페이지네이션 CSS 수정



- Pagination 동작 및 클래스 확인하기
- 각각 bullet 크기 설정
- 선택 효과 설정
  - 백그라운드 이미지 조절
  - 백그라운드 컬러 제거 → 투명하게 처리

# 이동 버튼 CSS 수정



- 버튼 위치부터 확인 → position: absolute; / top: 300px;
- 버튼 스타일 설정
  - Width / height : 55px;
  - Border: 2px solid #333;
  - Color: #333;
  - Border-radius: 50%;
  - Cursor: pointer;
- 화살표 크기도 수정하기 → after 의 폰트 사이즈 수정

# 이동 버튼 위치 지정



- Position: absolute; 는 이미 지정된 상태!
- Left: 400px? / right: 400px?
- 반응형 확인하기!
- Left: 50% / translate(-550px, 0);
- Right: 50% / translate(550px, 0);
- Promotion 크기를 기반으로 배치하기 → 단 일정 크기 부터는 반응형 필요!

# Promotion 에 overflow 지정



- Swiper 의 크기로 인해 가로 스크롤 발생!
- Promotion 에 overflow: hidden; 설정으로 스크롤 없애기!

# Autoplay?



- Pagination에 재생, 멈춤 버튼을 만드는 것은 고통 스러우니 공지 사항의 + 버튼에 autoplay 멈춤, 재시작 기능을 넣어 봅시다!
- <a> 태그에 onclick 주기!
- Js 파일에 함수 생성
- Swiper의 autoplay 관련 함수
  - .autoplay.start() / .autoplay.stop()
- Autoplay.stop() 적용 → 어? 그런데 안먹네요?



# Autoplay?

- <a> 태그의 특성상 href로 인해 페이지를 새로 고침
- 이를 해결하기 위해 href="javascript:함수명();" 을 사용
- A 태그의 효과는 쓰고 싶지만, 페이지 새로 고침을 막으려면
  - <a href="javascript:void(0);"> 사용

```
<a href="javascript:stopAutoPlay();">
    <span class="material-symbols-outlined">add_circle</span>
</a>
```

# Autoplay?



- swiperPromotion 의 값을 console.log 로 찍어보기
- swiperPromotion.autoplay.running 값에 따라 stop / start  
여부 결정!

# Promotion Toggle



- Promotion section 토글 기능 추가!
- Toggle 용 버튼에 클래스 주기!
  - Javascript:void(0); 처리
- JS 에서 버튼에 addEventListener 추가
  - Promotion 에 hide 클래스 추가 / 삭제 처리
- CSS 에서 promotion 처리

# Promotion Toggle 버튼 애니메이션!



- Promotion section 토글 버튼 애니메이션 추가하기!
- 보여 졌을 때 180deg 돌아가도록 처리하기!
- JS, CSS 수정!



스크롤에 따른  
애니메이션 처리!

# Scroll 발생을 체크



- 브라우저 레벨에서 발생하는 개념이므로
- **document → window** 사용!

```
// SCROLL
let scrollYpos;
window.addEventListener("scroll", function () {
  scrollYpos = window.scrollY;
  console.log(scrollYpos);
});
```

- 콘솔 로그에서 확인하기!

# 페이지가 로드되면 바로!



- Visual의 애니메이션 재생 필요
- Visual의 inner 클래스에 :hover로 처리했던 것으로 클래스로 변경 → :hover → animate
- 페이지가 로드 되면 바로 실행되는 함수
  - Window.onload() 사용

```
window.onload = () => {
  const visualInner = document.querySelector(".visual .inner");
  visualInner.classList.add("animate");
};
```

# 엘살바도르 애니메이션!



- 스크롤 위치 찾기! → 300 정도가 적당해 보이네요!
- Hover로 처리해 두었던 애니메이션을 animte 클래스로 변경!
- JS 처리

```
if (scrollYpos > 300) {  
    const elsalvadorAnimate = document.querySelector(".elsalvador");  
    elsalvadorAnimate.classList.add("animate");  
}
```



**JUST  
DID  
IT.**

