

Hello,

KDT 웹 개발자 양성 프로젝트
5기!

with



페이지
클론!

<https://radiant-maamoul-80f681.netlify.app/>



애니메이션 비쥬얼



실습, 이디오피아 구현하기



실습, Favorite 구현하기



실습, Magazine 구현하기

★ RESERVE MAGAZINE
리저브 매거진과 함께 길들이는 가을을 즐겨보세요.



실습, FIND STORE 구현하기



전국 어디에서나 스타벅스와 함께!
스타벅스와 함께 커피 한잔의 여유를 가져보세요.
나의 취향이 아름다운 콘, 스타벅스 라떼에 제일
더욱 편안하게 스타벅스를 만날 수 있는 드라이브 스루 패밀
한집에서 더 편안한 차이나와 소통 공간, 차운더티 스토어까지.
다양한 스타벅스 패밀리 여러분을 기다립니다.

마감 찾기



FOOTER



FOOTER 구현하기



COMPANY CORPORATE SALES PARTNERSHIP ONLINE COMMUNITY RECRUIT

한눈에 보기 단체 및 기업 구매 안내 신규 일정 계획 회사소개 재용 소개
스타벅스 사명 단체 주문 예약 안내 협력 고객사 충족신청 트위터 재용 지원하기
스타벅스 소개 국내 뉴스룸 블로그
국내 뉴스룸 세계의 스타벅스 유튜브 인스타그램
세계의 스타벅스 글로벌 뉴스룸

제인정부국민연금 | 영상보조디자인기기운증관련방침 | 출퇴근자 미증액금 | 스타벅스 카드 이용약관 | My DT Pass 서비스 이용약관 | 온라인영 원탁인
[제작하시는 팀] [신규집집제작] [서버트립]

Copied by letz from Starbucks Korea. © 2022 Starbucks Coffee Company. All Rights Reserved.

FOOTER Menu



```
<footer>
  <div class="inner">
    <!-- FOOTER MENU -->
    <div class="footer__menu">
      <div class="left">
        <ul class="menu">
          <li>
            <ul>
              <a href="#">COMPANY</a>
              <li>한눈에 보기</li>
              <li>스타벅스 사명</li>
              <li>스타벅스 소개</li>
              <li>국내 뉴스룸</li>
              <li>세계의 스타벅스</li>
              <li>글로벌 뉴스룸</li>
            </ul>
          </li>
        </ul>
      </div>
      <div class="right">
        
      </div>
    </div>
  </div>
```

FOOTER, menu 구현하기

- Flex 를 사용해서 메뉴 파트(left)와 로고 파트(right) 나누기
- 메뉴 파트는 넓이를 90%, 로고 파트는 10%로 부여

```
/* FOOTER MENU */  
footer .inner .footer__menu {  
    display: flex;  
}  
  
footer .inner .footer__menu .left {  
    width: 90%;  
}  
  
footer .inner .footer__menu .right {  
    width: 10%;  
}
```



실습, FOOTER 구현하기

COMPANY
한눈에 보기
스티박스 소개
스티박스 소개
국내 모스들
세계인 스티박스
글로벌 뉴스룸

CORPORATE SALES
단체 및 기업 구매 안내
단체 주문 배송 안내

PARTNERSHIP
신규 업종 개척
협력 고객사 등록신청

ONLINE COMMUNITY
쪽이스북
프리미
유튜브
인스타그램

RECRUITMENT
채용 소개
채용 자원화기

Footer

- .menu 의 li 크기를 20%으로 부여하여 left 90% 공간을 1/5로 분리하여 적용

FOOTER Sub-Menu



실습, sub_menu 구현하기

```
<div class="footer__sub-menu">  
    <ul>  
        <li><a href="#">개인정보처리방침</a></li>  
        <li class="contour"></li>  
        <li><a href="#">영상정보처리기기 운영 관리 방침</a></li>  
        <li class="contour"></li>  
        <li><a href="#">홈페이지 이용 약관</a></li>  
        <li class="contour"></li>  
        <li><a href="#">위치정보 이용 약관</a></li>  
        <li class="contour"></li>  
        <li><a href="#">스티박스 카드 이용 약관</a></li>  
        <li class="contour"></li>  
        <li><a href="#">비회원 이용 약관</a></li>  
        <li class="contour"></li>  
        <li><a href="#">My DT Pass 서비스 이용 약관</a></li>  
        <li class="contour"></li>  
        <li><a href="#">윤리경영 핫라인</a></li>  
    </ul>  
</div>
```

FOOTER button



실습, button 영역 구현하기

찾아오는 길
신규입점제의
사이트 맵

```
<!-- FOOTER_BTN -->  
<div class="footer__btn">  
    <a href="#" class="btn btn--white">찾아오는 길</a>  
    <a href="#" class="btn btn--white">신규입점제의</a>  
    <a href="#" class="btn btn--white">사이트 맵</a>  
</div>
```

실습, copylight 영역 구현하기



Cloned by tetz from Starbucks Korea. © 2023 Starbucks Coffee Company. All Rights Reserved.

```
<!-- FOOTER COPYLIGHT -->
<div class="footer__copylight">
  <p>
    Cloned by <span class="strong">tetz</span> from Starbucks Korea, ©
    2023 Starbucks Coffee Company. All Rights Reserved.
  </p>
</div>
```

수고하셨습니다!

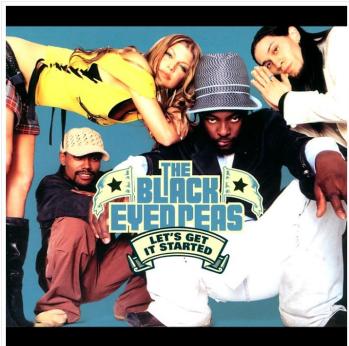
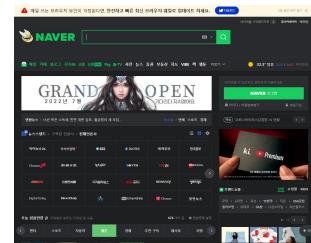


javascript!





WITHOUT



let str =
"이 글은 도커에 대해 1도 모르는 시스템 관리자나
서버 개발자를 대상으로 도커 전반에 대해 알고 넓은
지식을 담고 있습니다. 도커가 등장한 배경과 도커의
역사, 그리고 도커의 핵심 개념인 컨테이너와 이미지에
대해 알아보고 실제로 도커를 설치하고 컨테이너를 실
행해 보도록 하겠습니다.";

[이 글은 도커에 대해 1도 모르는 시스템 관리자나 서버 개발자를 대상으로 도커 전반에 대해 알고 넓은 지식을 담고 있습니다. 도커가 등장한 배경과 도커의 역사, 그리고 도커의 핵심 개념인 컨테이너와 이미지에 대해 알아보고 실제로 도커를 설치하고 컨테이너를 실행해 보도록 하겠습니다.]

• 위의 문장을 80byte 기준으로 잘라서 배열에 담기

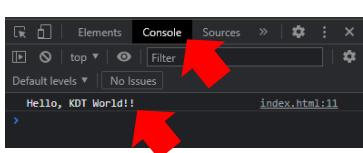
• 한글은 2byte, 한글을 제외한 영어, 숫자, 기호, 띄어쓰기는 1byte

• 자르고 난 다음 글자가 공백일 경우는 생략처리



```
<!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>
      console.log("Hello, JS World!");
    </script>
  </head>
  <body></body>
</html>
```

개발자 도구 실행!



```
<!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("Hello, JS World!");
    </script>
  </head>
  <body></body>
</html>
```

127.0.0.1:5599 내용:
Hello, JS world



3가지 CSS 참조 방식

인라인
방식



내장
방식



링크
방식



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

각각의 장단점이 존재 하겠죠?



• 내장 방식

- 간단하게 만들 수 있음
- 특정 페이지에서만 작동하는 기능일 경우 내장 방식으로만 따로 구현 가능

• 링크 방식

- JS 코드의 양이 많아지면 파일로 관리하는 편이 편함
- 같은 기능을 다른 페이지에서 사용하고 싶을 때 JS 파일 링크만 걸어서 사용 가능
- 유지 보수 용이성이 편리

일단 해봅시다!



• `Console.log("TEXT");`

- 우리의 디버깅 친구!
- 물론 이걸 쓰는건 안 좋은 방식입니다!

• `Alert("TEXT");`

- 화면에 뭐든 띠야 재미 있으니까!
- 자매품 `confirm("TEXT");`
- 얘는 취소 버튼이 있어요! → 값을 리턴한다 & IF문에 사용 가능!

실습

- 안녕하세요! `Console.log()` 로 띄우기
 - 단, 파일 링크 방식 사용
- 수업 언제 끝나나? `Alert()`로 띄우기
 - 단, 내장 방식 사용



읽기 순서?

- CSS의 방식 또는 선언 위치에 따라서 읽는 순서가 달라졌었습니다!
- JS에서도 확인해 볼시다!



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script>
      alert("제드 그린데 js 파일 링크 위")
    </script>
    <script src="./index.js"></script>
    <script>
      alert("제드 그린데 js 파일 링크 아래")
    </script>
  </head>
  <script>
    alert("애드枉 버디 사이");
  </script>
  <body>
    hello HTML World!
    <script>
      alert("버디 양쪽");
    </script>
  </body>
  <script>
    alert("버디 아래");
  </script>
</html>
```

Index.html

Index.js



읽기 순서!

- 말 그대로 읽히는 순서에 따라서 작동 합니다!!



JS 기초!



표기법

dash-case(kebab-case)
snake_case
camelCase
ParcelCase



thequickbrownfoxjumpsoverthelazydog



HTML CSS

dash-case(kebab-case)

the-quick-brown-fox-jumps-over-the-lazy-dog



HTML CSS

snake_case

the_quick_brown_fox_jumps_over_the_lazy_dog



JS

camelCase



theQuickBrownFoxJumpsOverTheLazyDog

JS

PascalCase

TheQuickBrownFoxJumpsOverTheLazyDog



생성자!

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 ~



```
let fruits = ['Apple', 'Banana', 'Cherry'];

console.log(fruits[0]); // 'Apple'
console.log(fruits[1]); // 'Banana'
console.log(fruits[2]); // 'Cherry'

console.log(new Date('2021-01-30').getDay()); // 6, 토요일
console.log(new Date('2021-01-31').getDay()); // 0, 일요일
console.log(new Date('2021-02-01').getDay()); // 1, 월요일
```

주석

Comments



```
// 한 줄 메모
/* 한 줄 메모 */

/***
 * 여러 줄
 * 메모1
 * 메모2
 */
```



단축키만 잘 외우시면 됩니다!



• 원도우

- Ctrl + /
- Ctrl + K + C / Ctrl + K + U

• 맥

- Cmd + /
- Cmd + K + C / Cmd + K + U

• 사용하시기 편한 방법으로 쓰시면 됩니다!

변수!





변수!

(Variable)



변수

데이터를 저장하고 참조(사용)하는 데이터의 이름
var, let, const



변수, let

```
// 재사용이 가능!
// 변수 선언!
let a = 2;
let b = 5;

console.log(a + b); // 7
console.log(a - b); // -3
console.log(a * b); // 10
console.log(a / b); // 0.4
```



변수, let

```
// 값(데이터)의 재할당 가능!
let a = 12;
console.log(a); // 12

a = 999;
console.log(a); // 999
```



변수, const

```
// 값(데이터)의 재할당 불가!
const a = 12;
console.log(a); // 12

a = 999;
console.log(a); // TypeError: Assignment to constant variable.
```



예약어

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

```
let this = 'Hello!'; // SyntaxError
let if = 123; // SyntaxError
let break = true; // SyntaxError
let .....;
```

break, case, catch, continue, default, delete, do, else, false, finally, for, function, if, in, instanceof, new, null, return, switch, this, throw, true, try, typeof, var, void, while, with, abstract, boolean, byte, char, class, const, debugger, double, enum, export, extends, final, float, goto, implements, import, int, interface, long, native, package, private, protected, public, short, static, super, synchronized, throws, transient, volatile, as, is, namespace, use, arguments, Array, Boolean, Date, decodeURI, decodeURIComponent, encodeURI, Error, escape, eval, EvalError, Function, Infinity, isFinite, isNaN, Math, NaN, Number, Object, parseFloat, parseInt, RangeError, ReferenceError, RegExp, String, SyntaxError, TypeError, undefined, unescape, URIError ...

JS의 데이터 종류!

데이터 종류(자료형)

String
Number
Boolean
Undefined
Null
Object
Array

String 문자형 데이터

```
// String 문자형 데이터  
// "" 따옴표를 사용  
  
let myName = "TETZ";  
var email = "xenosign@naver.com";  
let hello = `Hello ${myName}!`;  
  
console.log(myName);  
console.log(email);  
console.log(hello);
```

TETZ	index.js:10
xenosign@naver.com	index.js:11
Hello TETZ?!	index.js:12
>	



Number

숫자형 데이터

```
// Number 숫자형 데이터  
// 정수 및 부동소수점 숫자를 나타냄  
let number = 123;  
let opacity = 0.7;  
  
console.log(number);  
console.log(opacity);
```

123	index.js:17
0.7	index.js:18



Boolean

참, 거짓 데이터

```
// Boolean 참, 거짓 데이터  
// true, false 두 가지 값만 가지는 데이터  
let checked = true;  
let isShow = false;  
  
console.log(checked);  
console.log(isShow);
```

true	index.js:25
false	index.js:26



Undefined

미할당 데이터

```
// Undefined  
// 값이 할당되지 않은 상태를 표기  
let undef;  
let obj = {  
    abc: 123,  
};  
  
console.log(undef);  
console.log(obj.abc);  
console.log(obj.efg);
```

undefined	index.js:35
123	index.js:36
undefined	index.js:37



Null

의도된 빈 데이터

```
// Null  
// 어떤 값이 “의도적”으로 비어 있음을 의미할 때 사용  
let empty = null;  
  
console.log(empty);
```

null	index.js:43
------	-------------



Array

배열 데이터

```
// Array 배열 데이터  
// 여러 데이터를 순차적으로 저장  
let fruits = ["Pen", "Pineapple", "Apple", "Pen"];  
  
console.log(fruits[0], fruits[1], fruits[2], fruits[3]);
```

Pen Pineapple Apple Pen



Object

여러 데이터 꾸러미

```
let data = [1, "Apple", false, null, undefined];  
  
console.log(data[0], data[1], data[2], data[3], data[4]);
```

1 'Apple' false null undefined



```
// Object 객체형 데이터
// 여러 데이터를 key : value 의 형태로 저장

let tetz = {
  name: "이효석",
  age: "?",
  isOld: true,
  isMarried: false,
  hoby: ["Bicycle", "LOL", "Drink", "Travel"],
  brain: null,
  girlFriend: undefined,
};

console.log(tetz.name);
console.log(tetz.age);
console.log(tetz.isOld);
console.log(tetz.isMarried);
console.log(tetz.hoby);
console.log(tetz.brain);
console.log(tetz.girlFriend);
```



```
이효석
??
true
false
» (4) ['Bicycle', 'LOL', 'Drink', 'Travel']
null
undefined
```

실습

- 각자를 소개할 수 있는 Object 형태의 변수 선언
- 지금 까지 배운 모든 데이터를 사용하여 자신을 소개하는 console.log 만들기!



typeof



자료형을 알려주는 typeof



- 해당 자료형이 어떤 것인지 알려주는 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
```

자료형을 알려주는 typeof



- 변수가 아닌 친구에게도 쓸 수 있습니다!

```
console.log(typeof "이효석");
console.log(typeof 3);
console.log(typeof null);
```

```
string
number
object
```

실습

- Typeof 를 사용해서 아래의 문구를 만드세요
- "" 안의 문구는 typeof 의 결과값으로 출력해야 합니다.
- 출력 문구
 - "number" isn't "string" data type.
 - typeof 를 "boolean" 이나 "null" 에 사용하면, "object" 결과를 얻을 수 있습니다.



문자와 변수를 동시에!



문자 + 변수를 동시에 쓰고 싶을 때!



- 폐소드의 매개 변수로 넣어서 사용
 - `Console.log("문자", 변수, "문자");`
- 연산자를 사용해서 변수를 문자로 변환 후 더하여 사용
 - `Console.log("문자" + 변수 + "문자");`
- 뱠텍 문자 사용
 - '문자를 쓰다가 변수를 쓰고 싶으면 \${variable}처럼 쓰면 됩니다'

문자와 변수를 혼용해서 쓰고 싶을 때!



```
let nameArr = ["구슬기", "김계환", "김민정", "김범석"];  
  
let nameArr = ["구슬기", "김계환", "김민정", "김범석"];  
  
console.log("1 번째 이름은", nameArr[0], "입니다");  
console.log("2 번째 이름은", nameArr[1], "입니다");  
console.log("3 번째 이름은", nameArr[2], "입니다");  
console.log("4 번째 이름은", nameArr[3], "입니다");  
  
1 번째 이름은 구슬기 입니다  
2 번째 이름은 김계환 입니다  
3 번째 이름은 김민정 입니다  
4 번째 이름은 김범석 입니다
```

문자와 변수를 혼용해서 쓰고 싶을 때!



```
for(let i = 0; i < nameArr.length; i++) {  
    console.log(` ${i+1} 번째 이름은 ${nameArr[i]} 입니다`);  
}  
  
1 번째 이름은 구슬기 입니다  
2 번째 이름은 김계환 입니다  
3 번째 이름은 김민정 입니다  
4 번째 이름은 김범석 입니다
```

형 변환



성적을 구하는 프로그램 만들기



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

JS의 자동 형변환!

- 처음에는 편할 수도 있지만 큰 문제를 일으키게 됩니다.
- 방금도 Error 가 떴으면 바로 문제를 수정 했겠지만, Error 가 뜨지 않고 프로그램이 구동이 되었죠!
- 지금은 작은 프로그램이라 문제가 없었지만, 프로그램이 더 커진다면 의도하지 않았지만 정말 종대한 문제를 일으킬 수도 있습니다.
- 만약, 비트코인 거래 사이트라면?



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

명시적 형변환

- Bool 타입으로 변환 → `Boolean()`:

```
// Boolean 타입으로 변환  
console.log(Boolean(1));  
console.log(Boolean("이효석"));  
  
console.log(Boolean(0));  
console.log(Boolean(""));  
console.log(Boolean(null));  
console.log(Boolean(undefined));  
console.log(Boolean(NaN));
```

true
true
false



명시적 형변환 – 주의 사항

```
// 주의 사항  
console.log(Boolean(0));  
console.log(Boolean("0"));  
  
console.log(Boolean(""));  
console.log(Boolean(" "));
```

false
true
false
true
>

실습!

- 변수 `mathScore`, `engScore` 를 만들어 주세요.
- `mathScore = "77"; engScore = "88";`
- 시험 점수 평균을 계산하여 `avgScore` 에 저장하고, 이를 출력하는 프로그램을 작성하세요!
- 명시적 형 변환을 사용하여 평균 점수가 정확하게 나와야 합니다!



Variable



Variable



```
// var  
var name = "tetz";  
var name = "LHS";  
  
console.log(name);
```

LHS

```
// let  
let name = "tetz";  
let name = "LHS";  
  
console.log(name);
```

✖ Uncaught SyntaxError: Identifier 'name' has index.js:78
already been declared (at index.js:78:5)

```
// const  
const name = "tetz";  
const name = "LHS";  
  
console.log(name);
```

✖ Uncaught SyntaxError: Identifier 'name' has index.js:78
already been declared (at index.js:78:5)

“이불 밖은 위험해”



```
// var  
var tetz = "LHS";  
  
if (tetz == "LHS") {  
    var result = true;  
} else {  
    var result = false;  
}  
  
console.log(result);
```

true



```
// let  
var tetz = "LHS";  
  
if (tetz == "LHS") {  
    let result = true;  
} else {  
    let result = false;  
}  
  
console.log(result);
```

```
● Uncaught ReferenceError: result is not defined  
at index.js:9:13
```



var 의 문제점

- 중간의 같은 이름의 변수를 다시 선언해도 기존의 변수에 덮어 씌움
- 변수를 선언 했다는 건 분명이 다른 데이터를 넣으려는 것인데, 그것 을 기존의 데이터에 덮어 씌우면!? → 문제 발생!
- 그리고 변수가 {블록 단위}에서 끝나는 것이 아니라, 자기 맘대로 전 역으로 돌아다니고 영향력을 행사함 → 의도치 않은 문제 발생!
- 게다가 Debug 도 무지하게 어려움!
- 따라서 ES6 문법 부터는 var 대신 let 사용을 권장



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

- $num = num + 5 \rightarrow num += 5$
- $num = num - 5 \rightarrow num -= 5$
- $num = num * 5 \rightarrow num *= 5$
- $num = num / 5 \rightarrow num /= 5$



증가, 감소 연산자

- Num++;

- Num--;

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

10
11



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



논리 연산자



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, 남자이다.



FALSE

스티브 잡스는 한국인 이고 AND, 남자이다.

평가

OR는 첫번째 true를 발견하는 즉시 평가를 멈춤

스티브 잡스는 남자 이거나 OR, 한국인 이거나, 군인 이거나...

평가

AND는 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, ||

통과

```
// 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");
}
```



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, !

돌아가

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

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



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");
}

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

통과



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 쌤으면 → 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 = new Date().getDay();
switch (day) {
    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);
```

월요일



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 🙁");
```

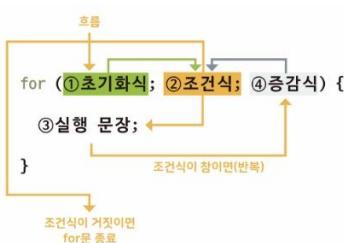
실습

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



반복문

For, while



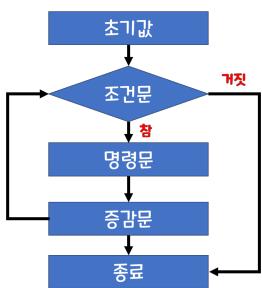
For 문

```
• 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 총접해서 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
상위 for 문입니다!, 1 번째 실행중
상위 for 문입니다!, 2 번째 실행중
상위 for 문입니다!, 3 번째 실행중
상위 for 문입니다!, 4 번째 실행중
상위 for 문입니다!, 5 번째 실행중
상위 for 문입니다!, 6 번째 실행중
상위 for 문입니다!, 7 번째 실행중
상위 for 문입니다!, 8 번째 실행중
상위 for 문입니다!, 9 번째 실행중
상위 for 문입니다!, 10 번째 실행중
상위 for 문입니다!, 11 번째 실행중
상위 for 문입니다!, 12 번째 실행중
상위 for 문입니다!, 13 번째 실행중
상위 for 문입니다!, 14 번째 실행중
상위 for 문입니다!, 15 번째 실행중
상위 for 문입니다!, 16 번째 실행중
상위 for 문입니다!, 17 번째 실행중
상위 for 문입니다!, 18 번째 실행중
상위 for 문입니다!, 19 번째 실행중
상위 for 문입니다!, 20 번째 실행중

```

실습



- 구구단을 반복문을 이용해서 console.log 로 출력해 보자!
- 변수를 console.log 로 출력하는 방법


```
let number = 10;
console.log("숫자는", number, "입니다");
console.log("숫자는" + number + "입니다");
```

숫자는 10 입니다
숫자는10입니다
- “콤마”를 사용하면 각각의 변수의 데이터 타입을 지키면서 출력
- “+”를 사용하면 숫자를 자동으로 문자로 변경 후, 문자열로 더하여 출력

실습



- 0 ~ 100의 숫자 중에서 2 또는 5의 배수 총합 구하기!

- 힌트
 - 나머지 연산자 % 를 사용
 - 5 % 3 → 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 높후!
- 주의하여 사용 필요



```

// while 문
// 1번 탑입, 조건문을 사용
let index = 0;

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

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

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

```



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



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} 번입니다!`);
```

```
1 번입니다!
3 번입니다!
5 번입니다!
7 번입니다!
9 번입니다!
```

실습!

- 1부터 1000까지의 숫자 중에서 짝수의 합을 구하는 프로그램을 만 들어 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`

// 함수 선언!
`function sum(a, b) { // a와 b는 매개변수(Parameters)
 return a + b;
}

// 재사용!
let a = sum(1, 2); // 1과 2는 인수(Arguments)
let b = sum(7, 12);
let c = sum(2, 4);

console.log(a, b, c); // 3, 19, 6`

// 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 사용해야 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 square(num) {  
    return num ** 2;  
}  
  
let squareNum = square(3);  
console.log(squareNum);
```

9

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!

- 삼각형의 넓이를 구하는 함수를 만들어 봅시다!
- 원의 넓이를 구하는 함수를 만들어 봅시다!
- 피타고라스의 정리를 이용해서 직각 삼각형의 밑변과 높이를 알 때, 빗변의 길이를 구하는 함수를 작성해 봅시다!
 - 루트를 씌우는 법 → $\text{Math.sqrt}(9) = 3;$

함수!

선언 방법들

함수 선언문 vs 함수 표현식

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

함수 선언문

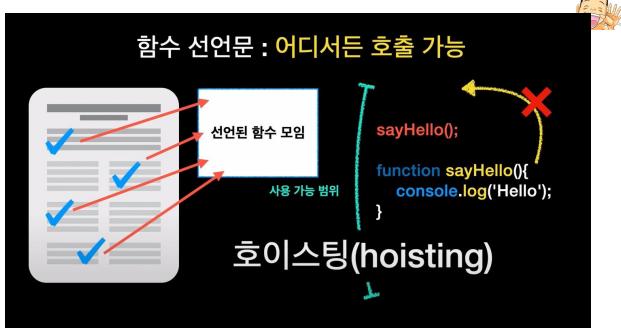
함수 선언문 vs 함수 표현식

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

함수 표현식

함수 선언문 : 어디서든 호출 가능

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



함수 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(arrow 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, Test

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>

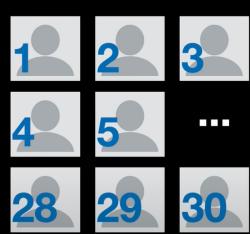
function test() {
  alert("TEST");
}
```



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

1번에 철수

2번에 영희

...

30번에 영수

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

index

0 1 29

```
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



```
let helloTest = "H-e-l-l-o";
let helloArr = helloTest.split("");
// .split: 문자를 인수 기준으로 끊어서 배열로 반환
// .reverse: 배열의 순서를 뒤집어서 반환
// .join: 배열을 인수 기준으로 문자로 병합해서 반환

console.log(helloArr);

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

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

let helloAgain = reverseHello.join("");
console.log(helloAgain);

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) 값**을 가지고 있는 경우에만 사용이 가능!
- `hello.split("")` → `['H', 'e', 'l', 'l', 'o']`라는 배열이 반환됨
- 배열에는 `reverse()`라는 메소드가 존재
- `hello.split("").reverse()` 는 `['H', 'e', 'l', 'l', 'o'].reverse()` 와 동일한 것!
- `['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

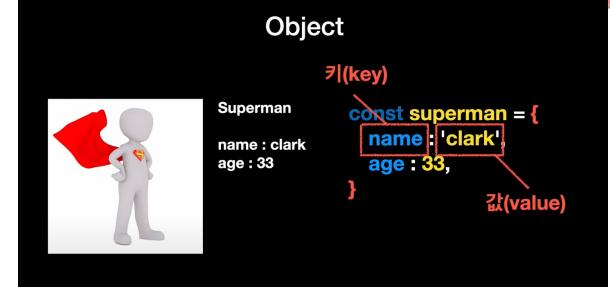
실습!



- kdt 배열의 제일 처음에 `tetz` 추가하기!
- 여기서 부터는 메소드 체이닝으로 처리 하시면 됩니다!
- kdt 배열의 제일 처음에서 `tetz` 를 빼고, 뺀 데이터를 글자별로 나누어 배열로 만들고, 해당 배열을 뒤집은 뒤, 다시 문자로 변환하여 출력하기!
- `shift()` 를 사용하면 빠진 데이터가 반환 됩니다!

 - ex. `let arr = ["tetz", "pororo"];`
 - `arr.shift();` → "tetz"

객체! Object



Object - 접근, 추가, 삭제

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);
```



실습!

- pororo라는 객체형 변수를 선언
- name: "뽀로로" / age: 7
- pororo의 이름과 나이를 출력하기
- pororo의 성별(gender)을 추가하고 출력하기
- Pororo의 키(height)를 추가하고 출력하기
- pororo의 성별 데이터를 삭제하고, pororo 객체를 출력하기

Object - 프로퍼티 존재 여부 확인

```
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!

- pororo가 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: function(){  
        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
Window {window: Window, self: Window, document: document, name: '', location: Location
n, -}
Hello, I'm
Window {window: Window, self: Window, document: document, name: '', location: Location
n, -}

Object



화살표 함수는 일반 함수와는 달리 자신만의 this를 가지지 않음

화살표 함수 내부에서 this를 사용하면, 그 this는 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

키(key)
값(value)

Superman
name : clark
age : 33

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



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 3명에 대한 정보를 생성자 함수를 이용하여 4개의 변수로 저장하세요.
- 참여자의 이름과 성별을 출력하는 method를 이용 5명의 정보를 출력해 보세요.

DOM

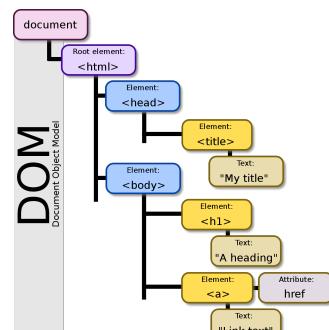
(Document Object Model)



DOM(Document Object Model)



- HTML 문서 요소의 집합!
- HTML 문서는 각각의 node와 object의 집합으로 문서를 표현
- 따라서 각각 node 또는 object에 접근하여 문서 구조 / 스타일 / 내용 등을 변경 할 수 있도록 하는 것!



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("요소 선택자")



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

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

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");
})
```

```
// 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
```

classList.add / remove / contains



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

```

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

```



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!



dom.html

```

<!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>

```

dom.js

```

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

```

null



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>

```

```

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>
  <script defer src=".//dom.js"></script>
</head>

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

```

```

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>
```



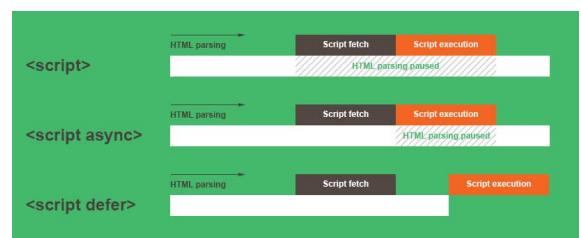
```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <div class="box">Box!!</div>
  <script src="./main.js"></script>
</body>
</html>
```



```
<!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



전체 관리하기!



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

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

// 첫 번째 메개변수(boxEl): 반복 중인 요소,
// 두 번째 메개변수(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);
});
```



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);
```

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, 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]

* NodeList(7) [div.box.box_1, div.box.box_2, div.box.box_3, div.box.box_4, div.box.box_5, 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]
```

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

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

1
KDT

실습



- 최 상단에 버튼(아무 요소나 가능) 하나 만들기
- 100 개의 box <div> 요소 만들기
- 버튼을 클릭하면 각각의 박스의 배경색이 변경 되는 페이지 만들기
- 첫번째 박스, 두번째 박스, 세번째 박스 ~ 열번째 박스 색상이 다르게 만들기
- 10개를 기준으로 색상이 반복 되도록 만들기

