JavaScript (1)

1장 변수 설명 전까진...

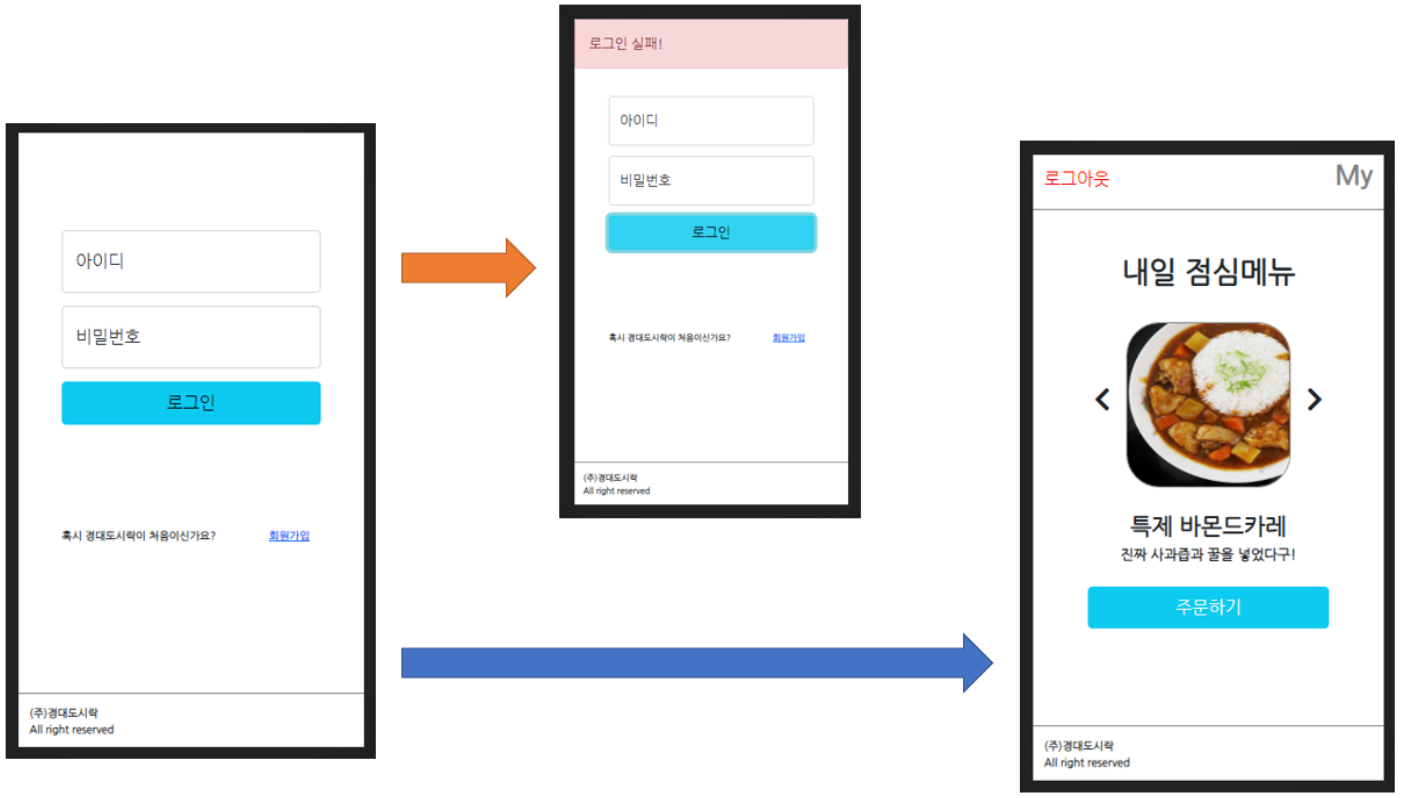
- JavaScript 가 어떤 언어인지 쭉 설명할 것이다.
- 어려운 개념을 엄청나게 설명할 것이지만,
- 그냥 마음 편하게 듣고
- 1장 변수설명할때부터 집중해서 듣길 바람

JavaScript는 프로그래밍 언어

- HTML Mark-up Language
- CSS Style-sheet Language

- JavaScript 브라우저에서 쓰이는 단 하나의 프로그래밍 언어
  - W3C에서 오직 JavaScript 만 표준으로 정해놓았다
  - 하나밖에 없으면 불편해도 배워야
  - 짧게 js 라고 한다
  - 정식 명칭은 ECMA Script
  - 모든 브라우저에 기본적으로 내장되어있음

JavaScript 가 있어야만 할 수 있는 일



로그인 버튼을 누르면,

1. 특정한 "함수" 실행됨
2. 입력받은 아이디, 패스워드를 서버로 보냄
3. 사용자 정보가 일치하면 메인화면으로
   일치하지 않으면 알림을 1초동안 띄웠다가 사라지게

Java 와 JavaScript

- 어디 가서 둘이 비슷한 언어라고 하지 말자.
- 완전히 다른 언어다.
  - 인도와 인도네시아만큼
  - 코끼리와 호랑이만큼
- Java, C, C++, Python, C# 중 그 어떤 언어도 JavaScript 와 비슷한 언어는 없다

근데 왜 JavaScript 인가? Java의 유명세에 따라가려고 이름지었을 뿐.

웹 개발자는 JavaScript를 잘해야한다

- JavaScript “방식” 을 잘 알고 있어야
- 코딩테스트에서 JavaScript 를 강조해서 본다면,

  C++, Java 로 풀 때와 완전히 다르게 접근해야한다.
- 면접에선 특히 2015년 이후의 JavaScript 를 아는지 볼 것
  - 흔히 ES6 라고 부르는 것

예를 들면 이런 문제를 낸다

- 1부터 n 까지의 수를 가진 배열이 있다.
  각 요소를 1씩 더해서 새로운 배열을 만들라.
  단, 반복문이나 재귀함수를 사용하지 말라

- 출제자의 의도:

  콜백함수를 쓸 줄 아는가
  정확하게는, map 함수를 쓸 줄 아는가

```
const datas = [1, 2, 3, 4, 5];
const newDatas = datas.map((data) => data + 1);
console.log(newDatas);
```

결과:

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

즉, JavaScript에서 함수는 곧 변수다.
다른 언어에선 함수는 변수가 아니다.

객체(Object)

- 다른 언어와는 다르게, 객체는 클래스의 인스턴스가 아니다.
- 심지어, 클래스는 다른 언어와는 설계와 동작이 완전히 다르다.

- JavaScript 는 엄청나게 독특하고 어려운 언어
- 이 수업에선 이미 C언어와 Python 의 기초를 안다고 가정한다
- 기초가 안 되어 있는 학생들을 위해, 자료실에 특별교안 링크를 올려두었다. (Notion)

배워볼 내용

- 1. 변수 let, const, var
- 2. 자료형(Type), 백틱(` `)
- 3. 조건문 if, else, ||, &&, ?, :
- 4. 배열(array)
- 5. 객체(object)
- 6. 함수 function, arrow function
- 7. 반복 callback, map(), filter()
- 8. 이벤트(Event) 와 돔(DOM) 조작
- 9. 비동기 통신 (async)

추가로, 7가지 간단한 게임 만들기

- 줌 수업에서 다루지 않는다. 두 가지 이유가 있다.
  1. 난이도가 매우 높음
  2. 다뤄야 할 양이 매우 많음
- 실습 줌 수업엔 포함시키지 않고, 자습할 수 있도록 영상 제공할 계획

1. 변수
const, let, var

결론만 말하자면,

- 일단 const 를 사용하라. (90% 의 상황)
- const 를 사용할 수 없는 상황엔 let 을 사용하라
- var 는 위험하다.

```
const sayHi = "안녕";
console.log(sayHi);
```

결과: 안녕

console.log() 는 프린트 함수라고 생각하자.

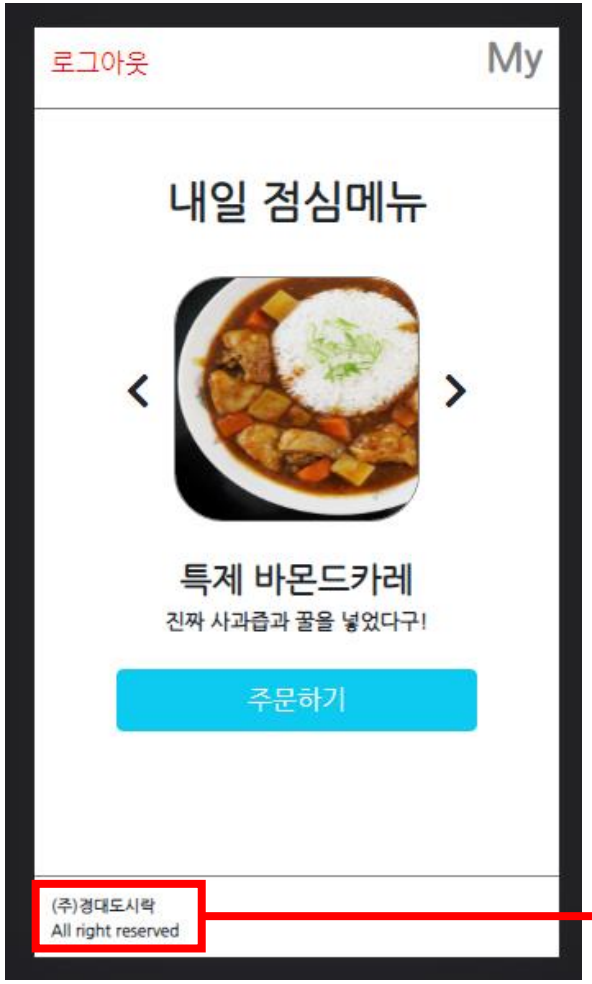
여러분이 마주칠 90% 의 상황에선, 변수선언은 const 로 충분하며, 가장 안전하다.

심화: const 에선 변수를 못바꾼다

```
> const sayHi = "안녕";
  sayHi = "헬로"; // error
✕ ▶Uncaught TypeError: Assignment to constant variable. VM24:2
      at <anonymous>:2:7
```

- const는 constant 의 줄임말이며, 즉 "상수"다
- 그러나, 완전한 상수는 아니고 상수 변수(constant variable)다.
- 예외:
  - 배열 element 변경, 추가 및 삭제
  - 객체 property 변경, 추가 및 삭제
- 나중에 배우겠지만, 이 두가지 예외 때문에 대부분의 선언은 const로 충분하다 – 90%의 상황

왜 변수를 안 바꾸는게 중요할까?



- immutability (불변성)
- 웹 개발자는 서버로부터 들어오는 데이터든, 내가 브라우저에서 만든 데이터든, 그 데이터를 유저가 신뢰할 수 있도록 만들어야
- 즉, 웹 개발자는 변하지 않을 데이터와, 변할 데이터를 구분해서 유저에게 제공할 줄 알아야

const는 개발자의 실수를 막아줄뿐만 아니라,
해커의 공격으로부터도 데이터를 보호해준다.

웹페이지 아래쪽에 있는 회사 정보는
이 페이지뿐만아니라
여러 페이지에서 쓰인다.
이 정보가 매번 바뀐다면, (주)경대도시락을 신뢰할 수 있는가?

실제 사례: 대구공고 해킹사건

인터넷 캡쳐 화면

const가 모든 걸 막아주진 않지만,
어느 정도는 막아준다.

쉽게 말하자면,

- 90% 의 변수선언은 const 를 쓰면 된다.
- 그럼 나머지 10% 는?

```
let sayHi = "안녕";
sayHi = "헬로";
console.log(sayHi);
```

결과: `'헬로'`

let 은 변수의 내용을 바꿀 수 있다.
단, 웹개발자가 let을 쓸 때는, 반드시 필요해서 써야한다.

var : 매우 위험

- 2015년 이전에 쓰이던 변수선언법
- 선언된 변수를 다시 선언해서 쓸 수 있다
- 프로그래밍 언어학적으로 매우 위험!
- 이걸 쓸 경우엔 기업 코딩테스트에서 떨어질 가능성이 크다.

```
var sayHi = "안녕";
var sayHi = "헬로";
console.log(sayHi);
```

결과: `'헬로'`

언어학적으로, 한번 선언한 변수는
동일한 이름으로 다시 선언하면 안 된다!

근데 왜 여전히 JavaScript에서 제공할까?
수억개의 사이트들이 var를 사용하여 코딩되었기때문
var를 제거했을 때 인터넷세계의 대혼란이 일어남

결론

- 일단 const 를 사용하라. (90% 의 상황)
- const 를 사용할 수 없는 상황엔 let 을 사용하라
- var 는 위험하다.

2. 자료형(type), 백틱(` `)

JavaScript 에서 쓸 수 있는 data type

number - 숫자를 의미.
0 ,1, 2, -1, -2.58, ...
string - '실'이라는 뜻인데, '문자열'을 의미한다.
"a", "hello", 'TypeScript', '0'
큰따옴표로 쓰던, 작은따옴표로 쓰던 똑같이 string이다.
따옴표를 생략하면 키워드나 변수로 인식하므로, 변수를 string으로 쓰고싶으면
반드시 따옴표를 써야 한다.
중요! 0을 따옴표로 감싸면 number가 아닌 string으로 인식한다.
boolean - true 또는 false를 말한다. 참과 거짓.
따옴표를 써서 'true'라고 쓰면 boolean이 아닌 string으로 인식한다.

array, object, function - 역시 자료형이나, 난이도가 있는 주제이므로 각각의 장에서
따로 다룬다.

* 키워드(keyword)는 특정한 작업을 하는 단어들을 말한다. 우리가 지금까지 배운 키워드는
const, let, var이며, 앞으로 더 많은 키워드들을 배워볼 것이다.

typeof 를 통해 내가 쓴 자료형이 무엇인지 확인해보자.

0과 "0" 은 다르다

```
const a = 0;
console.log(typeof a);  // number
const b = "0";
console.log(typeof b);  // string
```

비교연산자 === 로 확실히 알아보자

```
console.log(0 === "0");  // false
```

string 은 항상 따옴표 " " 를 써줘야한다

- 참고로, JavaScript 에선 String 이 아니라 string 이다

```
const a = david;  // error
```

여기서 david 는 string이 아니라 변수로 인식한다.
우린 david라는 변수를 선언한 적이 없다.
"david"라고 써주면 에러가 아니다.

boolean

- true 또는 false 를 말한다.

```
const a = true;
console.log(typeof a);  // boolean
```

- 하지만, “true” 라고 쓰면 string이다

```
console.log(typeof "true"); // string
```

백틱 ``

- 주로 문자열 안에서 변수나 함수 등을 좀 더 편하게 사용하려고 쓴다.
- 탭키 위에 작은따옴표 비슷하게 생긴 것이다.

백틱을 쓰지 않을 경우

```
const myName = "조교행님";
const age = 28;

console.log(myName + "은 " + age + "살이다.");
```

결과: `조교행님은 28살이다.`

백틱을 사용할 경우

```
const myName = "조교행님";
const age = 28;

console.log(`${myName}은 ${age}살이다.`);
```

${ } 안에 변수나 함수의 계산결과를 넣는다

+기호를 사용하지 않고도
문자열을 쭉 이어서 쓸 수 있다.

결과: 조교행님은 28살이다.

엔터와 탭 적용

그러나, 엄연히 말하면 결과는 다르다

```
console.log("조교행님");
console.log("나이: 28세");
console.log("사는곳: 이마트 옆");
```

```
console.log(`
    조교행님
    나이: 28세
    사는곳: 이마트 옆
`);
```

결과:

```
조교행님
나이: 28세
사는곳: 이마트 옆
```

결과:

```
조교행님
나이: 28세
사는곳: 이마트 옆
```

3. 조건문

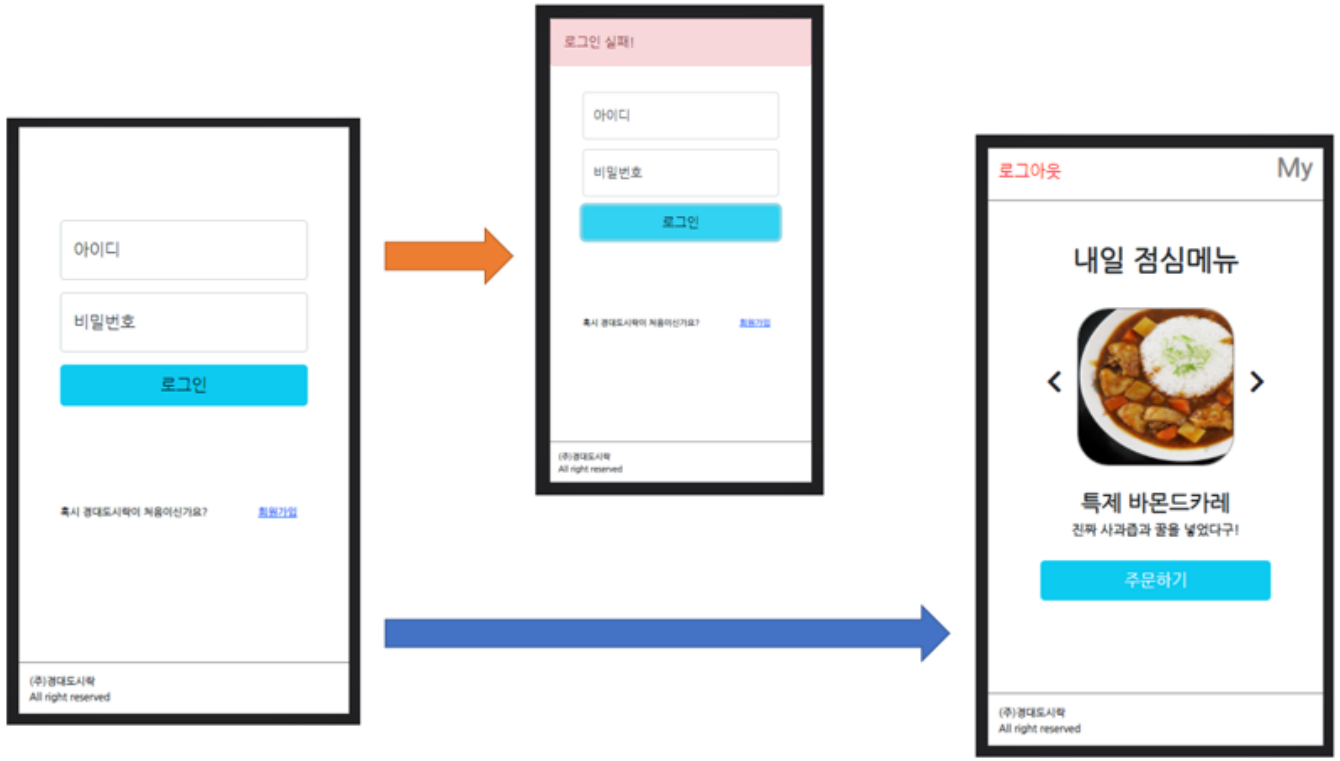
if, else

||, &&

?, :

대표적인 활용

- 로그인


아이디
비밀번호
로그인
로그인 실패!
아이디
비밀번호
로그인
로그아웃
My
내일 점심메뉴
특제 바몬드카레
진짜 사과즙과 꿀을 넣었다구!
주문하기

if else 는 c언어와 똑같은데,
== 대신 ===, != 대신 !== 를 사용하라

```
const myFavorite = "아빠";

if (myFavorite === "엄마") {
    console.log("엄마가 좋아");
} else if (myFavorite === "아빠") {
    console.log("아빠가 좋아");
} else if (myFavorite === "누나") {
    console.log("누나가 좋아");
} else {
    console.log("다 싫어");
}
```

왜냐면, == 또는 != 로만 쓰면 값 비교이기 때문이다.
=== 또는 !== 로 쓰면 데이터 타입까지 비교하므로 훨씬 정확하다.

```
console.log("0" == 0);  // true
console.log("0" === 0); // false

console.log("0" != 0);  // false
console.log("0" !== 0); // true
```

>, <, >=, <=

- 대소비교 역시 가능하다.

```
const age = 20;

if (age > 20) {
    console.log("술 판매 가능");
} else {
    console.log("영업정지 삐용삐용");
}
```

결과: 영업정지 삐용삐용

만약, 20세도 포함하고 싶다면 > 대신 >= 를 쓰면 된다.

조건 안에 조건도 가능하다

```
const myName = "조교행님";
const day = "일요일";

if (myName === "조교행님") {
    if (day === "일요일") {
        console.log("오늘은 출근 안함");
    } else {
        console.log("출근이다. 으악!");
    }
} else {
    console.log("누구냐 넌");
}
```

조건이 true 일 경우

- 다음은 기본이니 알고있을 것이다.

```
if (true) {
    console.log("실행");
}
```

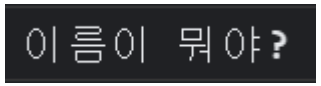
다음의 경우도 역시 true다.

```
const myName = "조교행님";
if (myName) {
    console.log("안녕");
} else {
    console.log("이름이 뭐야?");
}
```

결과: 

즉, JavaScript는 뭐라도 들어가있으면 true 로 받아들인다.

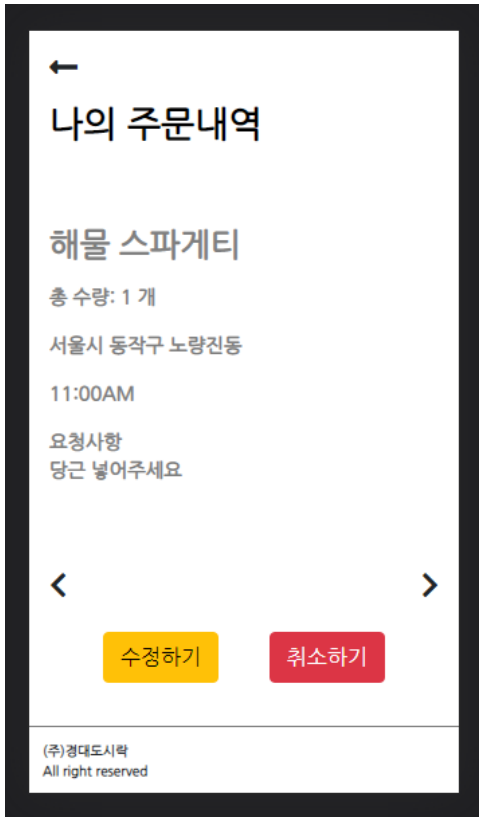
```
const myName = "";
if (myName) {
    console.log("안녕");
} else {
    console.log("이름이 뭐야?");
}
```

결과: 

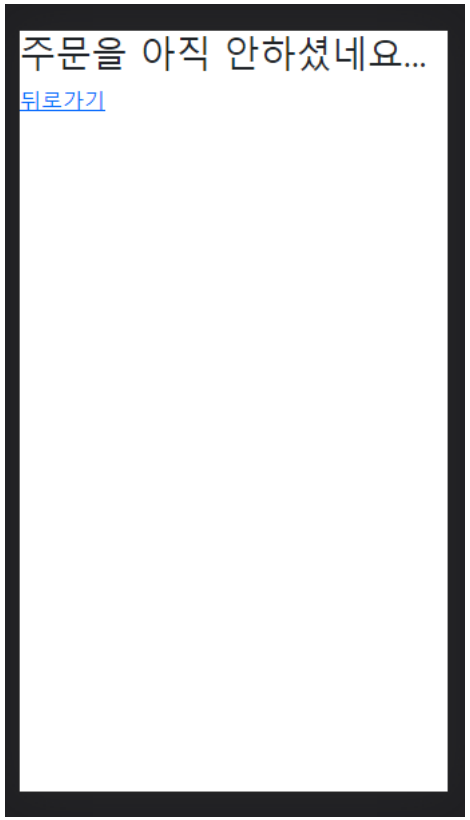
빈 문자열이기 때문에 false 이다.

- 조건문에서 무엇이 false 인지 아는것은 웹개발자에게 매우 중요하다
- 현재 데이터 상태에 따라,
  사용자에게 보여주는 화면을 다르게 만들 수 있기 때문

주문내역이 있을 경우



주문을 아직 안 했을 경우



비교에서 false로 인식하는 것

false	(boolean)
0	(number)
‘’ “”	(string. 빈 문자열)
undefined, null	(다음 장에서 배울 것)

- 추후 배우겠지만, 빈 배열 [ ] 과 빈 객체 { } 는 false 가 아니다!

- 위 언급한 것들을 제외하곤 모두 true 이다.