LIKELION UNIV. × HANYANG UNIV

FRONTEND SECTION

Coming soon

수업 준비중입니다

잠시만 기다려주세요...

LIKELION UNIV. × HANYANG UNIV

FRONTEND SECTION

3주차 수업

JavaScript - 기초

3주차

자바스크립트란?

정적 웹	vs	동적 웹
HTML		JavaScript
Java	**vs**	**JavaScript**

자바스크립트 실행해보기

개발자 도구에서 자바스크립트 실행해보기

맥북 - Cmd + Option + I

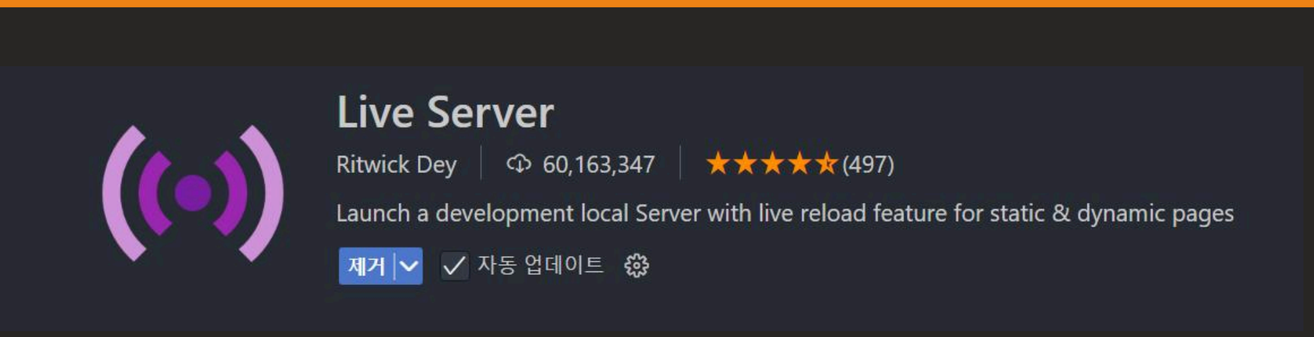
윈도우 - F12 or Ctrl + Shift + I

```
console.log('Hello JavaScript!');
```

자바스크립트 실행해보기

VScode에서 자바스크립트 실행해보기

1. vsCode의 확장 프로그램에서 Live Server 설치
2. 기존 HTML 코드가 있는 폴더에 Main.js 파일 생성(파일명은 상관 X)
3. HTML 코드 안에 js 코드 삽입
4. VScode 우측 하단에 Go Live 버튼 클릭 후 Chrome 창 뜨면 HTML 코드 클릭

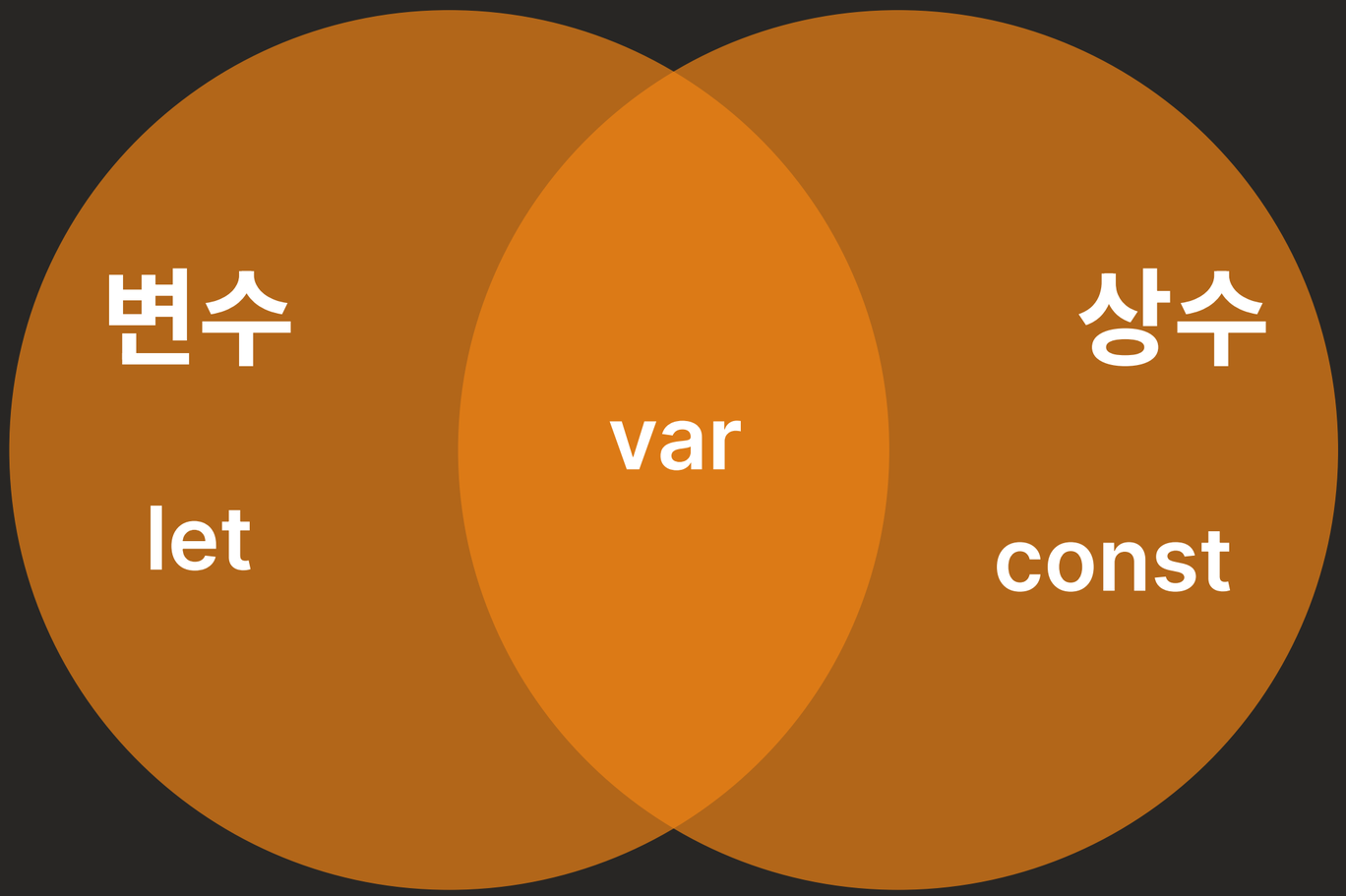


main.html > html

```
1   <!DOCTYPE html>
2   <html lang="ko">
3   <head>
4       <meta charset="UTF-8">
5       <title>JavaScript 실행 테스트</title>
6   </head>
7   <body>
8       <script src="main.js"></script>
9   </body>
10  </html>
```

변수와 상수

데이터를 저장하는 공간



선언, 할당, 초기화

```
선언키워드 상수이름;
상수이름 = 값;
선언키워드 상수이름 = 값;
```

```
let num;
num = 20;
const num = 10;
```

변수와 상수

데이터를 저장하는 공간

	let	var	const
변수 타입	변수	변수	상수
재할당	가능	가능	불가능
재선언	불가능	가능	불가능

자료형

데이터의 종류

숫자 (Number)	정수와 실수 구분하지 않음.
문자열 (String)	큰따옴표(“”), 작은 따옴표(‘’)
논리 (Boolean)	true(참), false(거짓) 두개 뿐
Undefined	값이 할당되지 않는 상태
Null	변수나 상수 선언 후 의도적으로 공간을 비워둘 때 사용
객체 (Object)	

자료형

데이터의 종류

객체 (Object)

배열

```
let A_score = [80, 70, 90, 60];
let B_score = [75, 90, 80, 85];
```

객체 리터럴

```
let A_score = {
  korean : 80,
  english : 70,
  math : 90,
  science : 60 };
```

```
let B_score = {
  korean : 75,
  english : 90,
  math : 80,
  science : 85 };
```

객체 구조 분해

객체의 속성을 추출하여 개별 변수로 할당하는 방법

type.js > ...

```
let A_score = {
    name : 'A',
    korean : 80,
    english : 70,
    math : 90,
    science : 60
};

let B_score = {
    name : 'B',
    korean : 75,
    english : 90,
    math : 80,
    science : 85
};

function print_score(score) {
    const {name, korean, science, math} = score;
    const text = name + '의 수학 점수는 ' + math + '이고 과학 점수는 ' + science + '이고 국어 점수는' + korean + '입니다'
    console.log(text);
};

print_score(A_score)
print_score(B_score)
```

연산자

어떤 연산을 처리하는데 사용하도록 미리 정의된 기호

산술 연산자

+ : 덧셈

- : 뺄셈

* : 곱셈

/ : 나눗셈

% : 나머지

** : 거듭제곱

비교 연산자

== : 값이 동일한지 비교, 데이터 유형 변환 가능

!= : 값이 다른지 비교, 데이터 유형 변환 가능

=== : 값과 데이터 유형이 모두 동일한지 비교

!== : 값 또는 데이터 유형이 다른지 비교

> : 왼쪽이 오른쪽보다 큰지 비교

< : 왼쪽이 오른쪽보다 작은지 비교

연산자

어떤 연산을 처리하는데 사용하도록 미리 정의된 기호

논리 연산자

&&

왼쪽부터 비교를 시작해

거짓인 것을 바로 반환

만약 모든 값들이 참이라면

가장 오른쪽 값 반환

||

왼쪽부터 비교를 시작해

참인 것을 바로 반환

만약 모든 값들이 거짓이라면

가장 오른쪽 값 반환

!

무조건 참인건 거짓으로

거짓인 건 참으로 반환

조건문

할까 말까 할까 말까 선택의 과정

If 문 | 만약 비가 온다면, 우산을 가져가라

```
if (비옴) {
    우산가져가기():
}
```

조건문

할까 말까 할까 말까 선택의 과정

else 문 | 만약 비가 온다면, 우산을 가져가라. 그렇지 않다면, 가벼운 옷차림을 하라

```
if (비옴) {
    우산가져가기():
} else{
    가벼운 옷차림();
}
```

조건문

할까 말까 할까 말까 선택의 과정

else if 문

만약 비가 온다면, 우산을 가져가라.
그렇지 않고 바람이 분다면, 자켓을 입어라
그 외에는 가벼운 옷차림을 하라

```
if (비옴) {
    우산가져가기():
} else if (바람분다){
    자켓 입기():
} else{
    가벼운 옷차림();
}
```

입력 받아보기

prompt(): 사용자에게 창을 띄워 데이터를 입력받을 수 있는 함수

```
prompt(message, default);
```

message: 사용자에게 표시될 메세지를 나타내는 문자열
default: 입력 상자에 표시될 기본값

반복문

지정한 조건이 참으로 평가되는 동안 지정한 블록문을 반복해서 실행

While 문

```
while (조건식){
    // 조건식이 참이면 실행
}
```

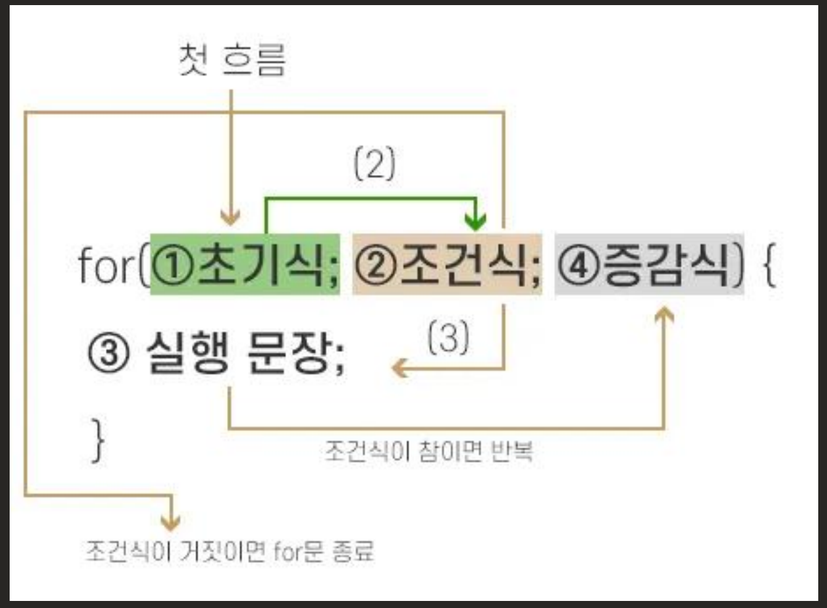
do ...while 문

```
do{
//블록문
}while(조건식);
```

반복문

지정한 조건이 참으로 평가되는 동안 지정한 블록문을 반복해서 실행

for 문




```
for (초기화식; 조건식; 증감식) {
// 실행될 코드 }
```

함수

특정 입력을 받아 일정한 처리를 수행한 후 결과를 반환하는 기능

함수 선언문

function 키워드로 함수를 정의하는 방법

fuction 키워드 뒤에 함수를 식별할 수 있도록 식별자만 붙여주면 됨

```
function 식별자(){
// 실행할 코드
}
```

```
fuction gugudan(){
    for(let i = 1;, i<=9, i++){
        console.log(`3 * ${i] = ${3 * i}`);
    }
}
```

함수

특정 입력을 받아 일정한 처리를 수행한 후 결과를 반환하는 기능

함수 표현식

함수를 변수에 할당할 수 있음

```
const gugudan = fuction gugudan(){
    for(let i = 1;, i<=9, i++){
        console.log(`3 * ${i] = ${3 * i}`);
    }
};  gugudan();  //함수 호출
```

함수

특정 입력을 받아 일정한 처리를 수행한 후 결과를 반환하는 기능

화살표 함수

화살표 함수는 익명 함수로만 정의 가능
함수 표현식을 사용해 정의

```
const gugudan = () => {
  for(let i = 1;, i<=9, i++){
     console.log(`3 * ${i] = ${3 * i}`);
  }
};

gugudan();  //함수 호출
```

Spread

객체나 배열을 펼쳐서 복사하거나 병합할 수 있음

`...` 을 사용하여 기존 객체를 건드리지 않고 새로운 객체나 배열을 만들 수 있다.

```
const lion = {
    name : '사자' };

const bravelion = {
    name : '사자',
    attribute : 'brave' };

const bravelikelion = {
    name : '사자',
    attribute : ' brave',
    color : 'yellow' };
```

```
const lion = {
    name : '사자' };

const bravelion = {
    ...lion,
    attribute : 'brave' };

const bravelikelion = {
    ...bravelion,
    color : 'yellow' };
```

rest

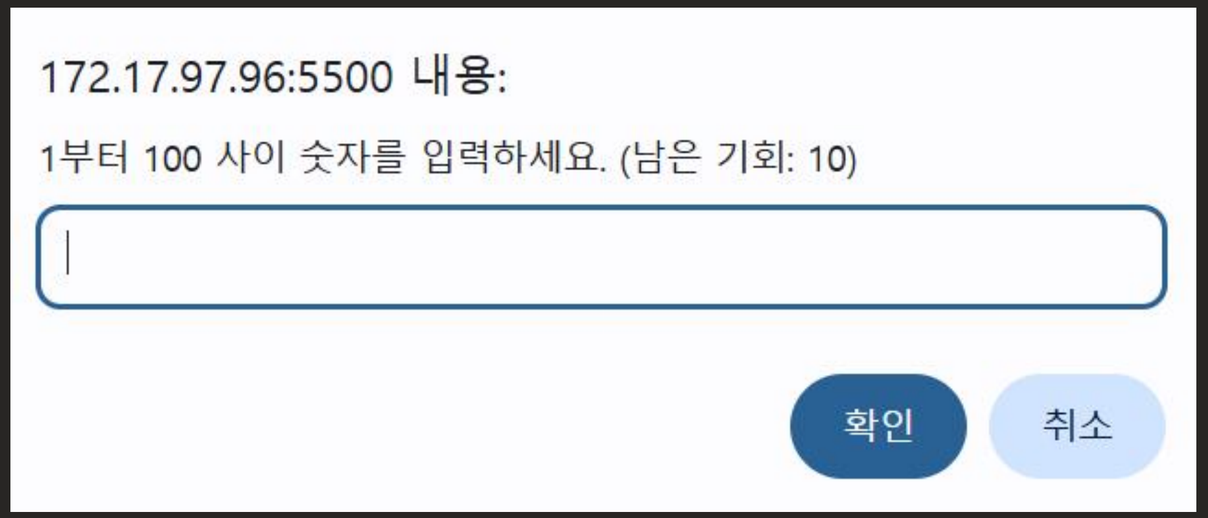
나머지 값들을 변수로 모아서 저장

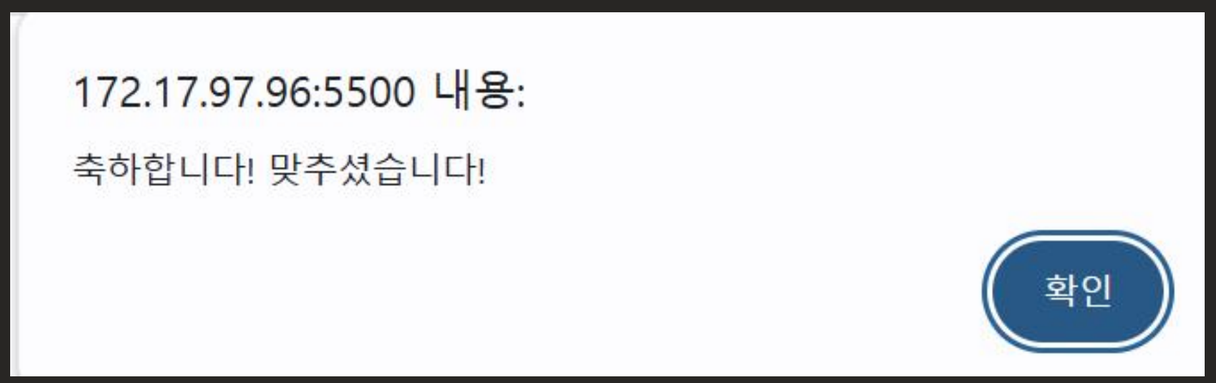
`...` 을 사용하여 기존 객체를 건드리지 않고 함수의 인자, 객체, 배열 등을 분해할 수 있다.

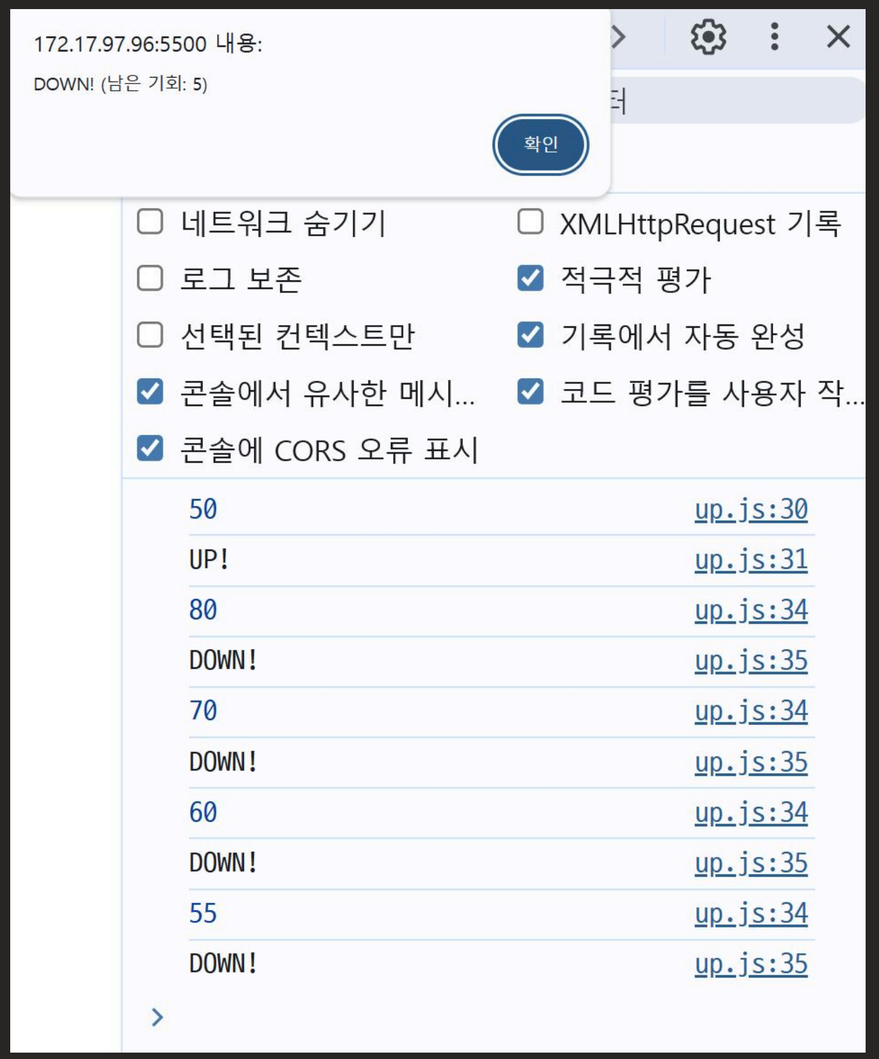
```
const bravelikelion = {
    name : '사자',
    attribute : ' brave',
    color : 'yellow' };

const {color, ...rest} = bravelikelion;
console.log(color);
console.log(rest);
```

과제


172.17.97.96:5500 내용:
1부터 100 사이 숫자를 입력하세요. (남은 기회: 10)
확인
취소


172.17.97.96:5500 내용:
축하합니다! 맞추셨습니다!
확인


172.17.97.96:5500 내용:
DOWN! (남은 기회: 5)
확인
네트워크 숨기기
XMLHttpRequest 기록
로그 보존
적극적 평가
선택된 컨텍스트만
기록에서 자동 완성
콘솔에서 유사한 메시...
코드 평가를 사용자 작...
콘솔에 CORS 오류 표시
50 up.js:30
UP! up.js:31
80 up.js:34
DOWN! up.js:35
70 up.js:34
DOWN! up.js:35
60 up.js:34
DOWN! up.js:35
55 up.js:34
DOWN! up.js:35

과제

1. 컴퓨터가 1부터 100 사이의 숫자 하나를 랜덤으로 정합니다.
2. 사용자는 그 숫자를 맞추기 위해 1부터 100 사이의 숫자를 입력해야 합니다.
3. 입력한 숫자가 정답보다 작으면 "UP!"이라는 메시지가 나옵니다.
4. 입력한 숫자가 정답보다 크면 "DOWN!"이라는 메시지가 나옵니다.
5. 정답을 맞추면 "축하합니다! 정답입니다." 라는 메시지가 나오고 게임이 끝납니다.
6. 만약 시도 횟수가 10번을 넘어간다면 “실패!” 라는 메시지가 나오고 게임이 끝납니다.

힌트

```
Math.floor(Math.random() * 100) + 1
 // 랜덤으로 1~100 사이의 정수를 뽑는 코드
```

조건

사용자가 맞았는지 틀렸는지

판별하는 함수를 만들 것