

프론트엔드 5주차

Java Script



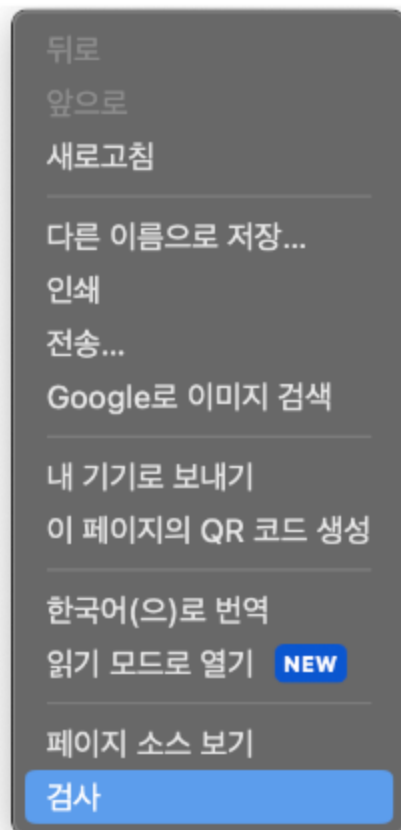


1	자바스크립트 기본	03
2	데이터 타입	06
3	연산자	10
4	조건문	13
5	반복문	20
6	함수	24

1. 자바스크립트 기본

파일 생성

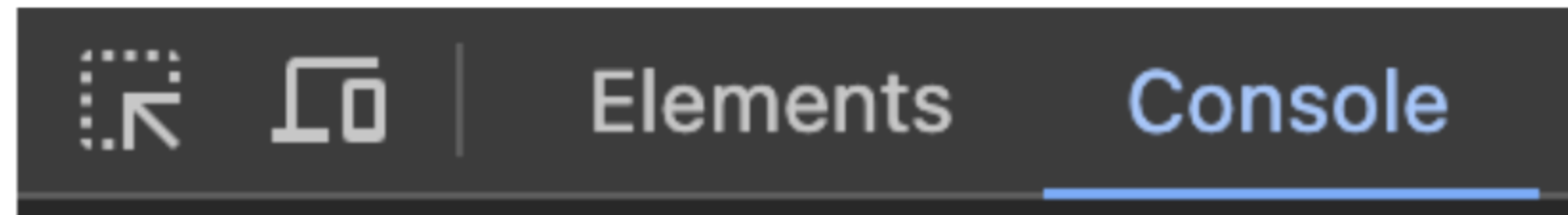
2. html파일 실행 -> 우클릭 -> 검사 클릭



1. script 태그

```
<script src="index.js"></script>|
```

3. 콘솔 진입



1. 자바스크립트 기본

변수

변할 수 있는 값. 즉 저장하고 싶은 공간.

VAR키워드



EXAMPLE

```
var hi;  
hi = '계명대 멋사';
```

```
console.log(hi);
```

콘솔에 출력되게 해주는 함수



OUTPUT

계명대 멋사

index.js:5

1. 자바스크립트 기본

변수

변할 수 있는 값. 즉 저장하고 싶은 공간.

var 키워드



EXAMPLE

```
var hi;  
hi = '계명대 멋사';  
  
console.log(hi);
```



```
hi = "12기"  
console.log(hi);
```



OUTPUT

계명대 멋사

index.js:5

12기

index.js:8

1. 자바스크립트 기본

상수

변경할 수 없는 값.

const 키워드

- 선언부에서 값을 할당 해야함.
- 변수 재할당 X



EXAMPLE

const hi;



```
index.js / ...  
1  const hi; 'const' declarations must be initialized.
```



const hi = "계명대 멋사";



재할당 X

```
const hi = "계명대 멋사";  
console.log(hi);
```

```
hi = "12기";  
console.log(hi);
```

계명대 멋사

index.js:2

```
✖ ▶ Uncaught  
    TypeError: Assignment to  
    constant variable.  
    at index.js:4:3
```

1. 자바스크립트 기본

변수, 상수 선언 규칙

- ✓ 첫 번째 글자는 영어, \$, _만 가능.
- ✓ 첫 글자 이외는 숫자도 가능
- ✓ 영어 대소문자 구별(abc ≠ ABC)
- ✓ var, const처럼 자바스크립트 예약어는 사용불가
- ✓ 공백 허용 X

2. 데이터 타입

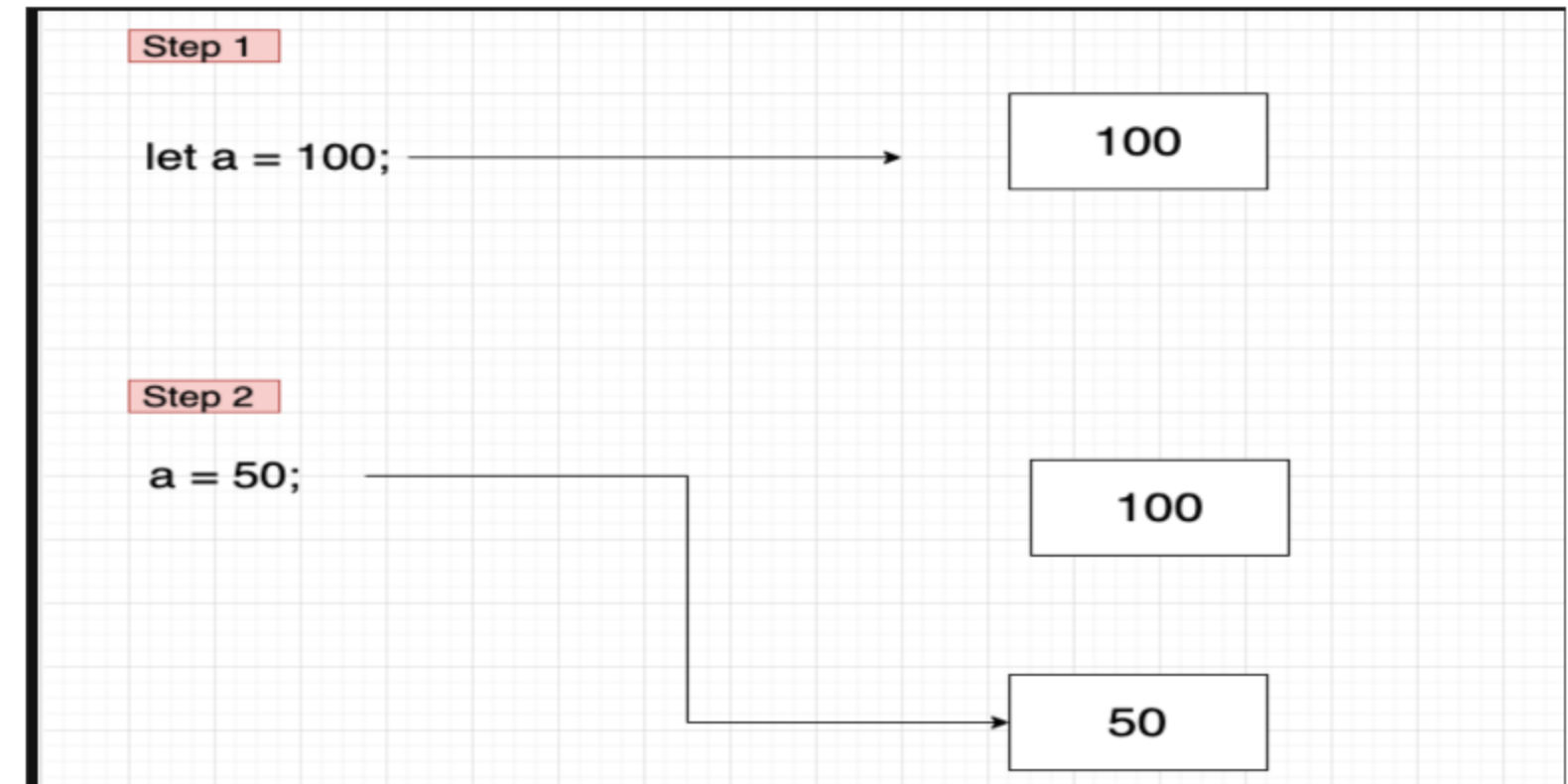
원시 데이터 타입

저장된 값을 변수가 가리키는 형태.
객체가 아니다.

- ✓ **String** - 문자열
- ✓ **number** - 숫자형
- ✓ **bigint** - 숫자형으로 표현될 수 없는 값
- ✓ **boolean** - true, false
- ✓ **undefined** - 값이 지정되지 않았다

✓ 원시타입 특징

불변성을 갖고 있기 때문에 기존에 메모리에 생성된 값들은 그 자체가 변경될 수 없다.



2. 데이터 타입

참조 데이터 타입

- 원시데이터를 제외한 모든 타입
- 주소를 저장



원시 vs 참조

- 원시 타입은 **직접적으로 값을** 가리키는 타입,
참조 타입은 **주소 값을** 할당
- 변수가 동적으로 변함



배열



객체



함수

2. 데이터 타입

참조 데이터 타입

- 원시데이터를 제외한 모든 타입
- 주소를 저장

- 배열
- 객체
- 함수



배열이란?

자료형의 나열, 데이터들의 집합



선언

```
const arr = [1,2,3];
```



배열 모양

0	1	2
1	2	3

3. 연산자

여러가지 연산자(1)

산술연산자 - 사칙연산, 나머지연산(%), 제곱(^)

증감연산자 - 숫자를 증가시킴

- ++: 1씩 증가
- --: 1씩 감소

비교연산자 - >, <, >=, <=

- ==: 추상 비교
- ===: 엄격 비교(type까지)
- !==: 양쪽이 다른지

3. 연산자

여러가지 연산자(2)

논리 연산자 - And, Or, Not

✓ AND 연산(&&) - 두 개의 조건이 모두 참이어야 참이 나옴

X	Y	X AND Y
TURE	TURE	TURE
TURE	FALSE	FALSE
FALSE	TURE	FALSE
FALSE	FALSE	FALSE

✓ OR연산(||) - 두 개의 조건 중 하나만 참이더라도 참이 나옴


X	Y	X OR Y
TURE	TURE	TURE
TURE	FALSE	TURE
FALSE	TURE	TURE
FALSE	FALSE	FALSE

✓ NOT연산(!) - 반대로 바꿈

X	NOT X
TURE	FALSE
FALSE	TURE

3. 연산자

여러가지 연산자(2)

 삼항연산자 - 조건의 결과가 참,거짓인지에 따라
수행조건이 달라짐

ex) $2 < 3 ? \text{'참'} : \text{'거짓'}$

4. 조건문

조건문

특정 조건 만족 시 실행하는 명령의 집합이며,
어떤 작업을 수행하고 싶을 때 사용하는 것



문법

```
if( 조건식 ){  
    참일때 실행  
} else {  
    거짓일때 실행  
}
```

if문

switch문

4. 조건문

조건문

특정 조건 만족 시 실행하는 명령의 집합이며,
어떤 작업을 수행하고 싶을 때 사용하는 것

if문

switch문

✓ 예시

```
const a =10;
const b =20;

if(a>b){
    console.log("a가 더 작다");
} else {
    console.log("b가 더 크다");
}
```

✓ 출력 값

b가 더 크다

[index.js:7](#)

4. 조건문

조건문

특정 조건 만족 시 실행하는 명령의 집합이며,
어떤 작업을 수행하고 싶을 때 사용하는 것

if문

switch문



else if문



```
const a =10;  
const b=20;  
const c=20;  
  
if(a>b){  
    console.log("a가 더 작다");  
} else if(b === c){  
    console.log("b랑c가 같다.")  
}else{  
    console.log("c가 크다");  
}
```

출력값: b랑 c가 같다.



코드 간결화

4. 조건문

조건문

특정 조건 만족 시 실행하는 명령의 집합이며,
어떤 작업을 수행하고 싶을 때 사용하는 것

if문

switch문



문법

```
switch (조건 값) {
    case 값1:
        조건 값이 값1일 때 실행하고자 하는 실행문;
        break;
    case 값2:
        //조건 값이 값2일 때 실행하고자 하는 실행문;
        break;
    default:
        조건 값이 어떠한 case 절에도 해당하지 않을 때 실행하고자 하는 실행문;
        break;
}
```

4. 조건문

조건문

특정 조건 만족 시 실행하는 명령의 집합이며,
어떤 작업을 수행하고 싶을 때 사용하는 것

if문

switch문

예시

```
const num = 10;

switch(num){
  case 1:
    console.log(num);
    break;

  default:
    console.log("아무것도 해당 안됨");
}
```

출력 값

아무것도 해당 안됨 [index.js:9](#)

4. 조건문

조건문

특정 조건 만족 시 실행하는 명령의 집합이며,
어떤 작업을 수행하고 싶을 때 사용하는 것

if문

switch문



break가 없을 때

```
const num = 10;

switch(num){
  case 10:
    console.log(num);

  case 2:
    console.log(num);
    break;
  default:
    console.log("아무것도 해당 안됨");
}
```



출력 값

10	<u>index.js:5</u>
10	<u>index.js:8</u>

4. 조건문

조건문

특정 조건 만족 시 실행하는 명령의 집합이며,
어떤 작업을 수행하고 싶을 때 사용하는 것



if문



switch문



예제

- 변수 num을 선언하고 num이 홀수인지 짝수인지 구분하는 조건문을 작성해 보세요

1. if문
2. switch문

4. 조건문

조건문

특정 조건 만족 시 실행하는 명령의 집합이며,
어떤 작업을 수행하고 싶을 때 사용하는 것

if문

switch문



예제

```
const num=10;

if(num % 2 ==0){
    console.log("짝수");
}else{
    console.log("홀수");
}

switch(num % 2){
    case 0:
        console.log("짝수");
        break;
    case 1:
        console.log("홀수");
        break;
}
```

5. 반복문

반복문

어떤 명령을 반복적으로 사용해야 할 때 사용.

for문

while문

✓ 문법

```
for( [초기문]; [조건문]; [증감문] ) {  
    문장;  
}
```

✓ 예시

```
for( let i=0; i<5; i++){  
    console.log(i);  
}
```

5. 반복문

반복문

어떤 명령을 반복적으로 사용해야 할 때 사용.

for문

while문



문법

```
while(조건문) {  
    문장;  
}
```



예시

```
while(i < 5) {  
    console.log("i = " + i);  
    i++;  
}
```

5. 반복문

반복문

어떤 명령을 반복적으로 사용해야 할 때 사용.

for문

while문

do-while문



- 조건을 끝 위치로 이동
- 한번은 무조건 실행
- 조건을 만족하지 못할때 루프 탈출



JavaScript ▾

```
let i = 0;  
  
do{  
    console.log(i++);  
} while(반복을 유지 할 조건, i < 10)
```


6. 함수

함수

하나의 동작, 원하는 결과를 도출하기 위한 코드의 집합

일반 함수

화살표 함수

✓ why?

- 재사용성, 모듈화

ex) 새우 볶음밥, 김치 볶음밥, 계란 볶음밥



볶음밥을 함수화

✓ 선언 방법

```
function 함수명(매개변수1, 매개변수2, -----){  
    실행코드  
}
```

6. 함수

함수

하나의 동작, 원하는 결과를 도출하기 위한 코드의 집합

일반 함수

화살표 함수

✓ 선언 방법

```
function 함수명(매개변수1, 매개변수2, -----){  
    실행코드  
}
```

✓ 실행 예시

```
//함수 선언  
function bok(main){  
    console.log(main + "볶음밥");  
}  
  
//함수 호출  
bok("새우");
```

6. 함수

함수

하나의 동작, 원하는 결과를 도출하기 위한 코드의 집합



선언 방법

```
//형식
const 함수명 = (파라미터) => {
    실행문
};
```



실행 예시

```
const sum = (a, b) => {
    return a+b;
};
```

일반 함수

화살표 함수

6. 함수

함수

하나의 동작, 원하는 결과를 도출하기 위한 코드의 집합

지역변수(로컬변수)

전역변수(글로벌 변수)



중괄호 안에서 선언 -> 중괄호 안에서만 사용가능

```
var a = 10 //전역변수

function test(hoho){
    var test1 = 1; // 중괄호 안에서 선언해서 지역변수
    console.log(test1);
    console.log(a);
}

console.log("지역변수 확인 : " + test1) // 지역변수 사용
```



✖ ▶ Uncaught ReferenceError: test1 is not defined
at [index.js:9:28](#)

6. 함수

함수

하나의 동작, 원하는 결과를 도출하기 위한 코드의 집합

지역변수(로컬변수)

전역변수(글로벌 변수)



중괄호 밖에 선언 -> 전체 사용 가능

```
var a = 10

function test(hoho){
    var test1 = 1;
    console.log(a);
}

test()
console.log("전역변수 확인 : " + a)
```



10

전역변수 확인 : 10

6. 함수

함수

하나의 동작, 원하는 결과를 도출하기 위한 코드의 집합



지역변수(로컬변수)



지역, 전역 변수 중복 선언



지역, 전역 중복선언

```
var a = 10

function test(hoho){
    var a = 1;
    console.log(a);
}

test()
console.log("전역변수 중복 확인 : " + a)
```



1

전역변수 중복 확인 : 10

6. 함수

함수

하나의 동작, 원하는 결과를 도출하기 위한 코드의 집합



구구단 함수 만들어보기

$$4 \times 0 = 0$$

$$4 \times 1 = 4$$

$$4 \times 2 = 8$$

$$4 \times 3 = 12$$

$$4 \times 4 = 16$$

$$4 \times 5 = 20$$

$$4 \times 6 = 24$$

$$4 \times 7 = 28$$

$$4 \times 8 = 32$$

$$4 \times 9 = 36$$

$$4 \times 10 = 40$$

6. 함수

함수

하나의 동작, 원하는 결과를 도출하기 위한 코드의 집합



구구단 함수 만들어보기

//특정 구구단 함수 만들기

```
function gugu(n){
    for(var i =0; i<=10; i++){
        console.log(n + "X" + i + "=" + n*i)
    }
}

gugu(4)
```




THANK YOU

함께해주셔서 감사합니다

