# 기초

## Javascript 기초

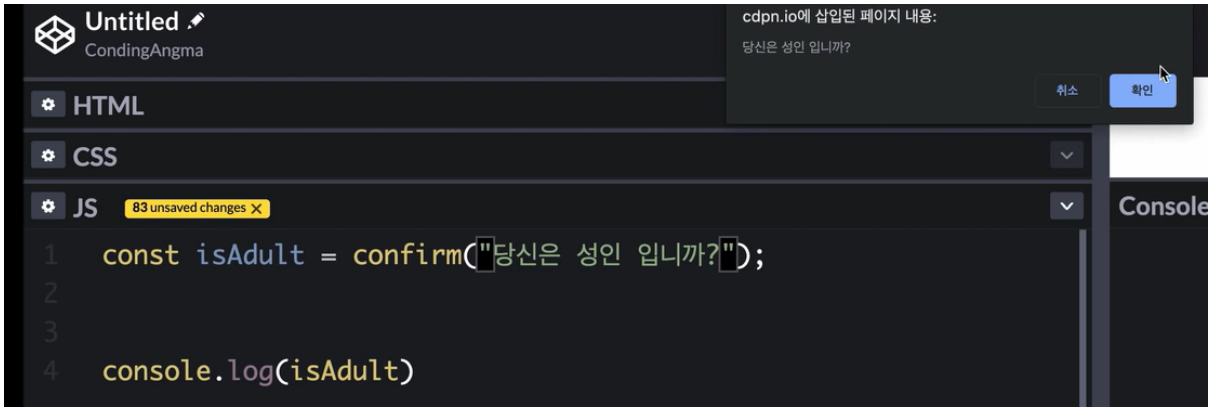
- 변수 선언
  - **let**: 변할 수 있는 값
  - const: 변하지 않는 값
  1. 변수는 문자, 숫자, $, _만 사용 가능
  2. 첫 글자는 숫자가 될 수 없음
  3. 예약어는 사용할 수 없음
  4. 상수는 대문자로 알려 주기 예) const MAX_SIZE = 99;
  5. 변수명은 읽기 쉽고 이해할 수 있게 선언
  - 앞에 \을 넣어 주면 특수 문자 인식
  - ${변수}, ${표현식}
- 숫자는 사칙연산 가능
- typeof 연산자: 변수의 자료형을 알아냄 예) console.log(type of "안녕")
- 형 변환 (**대문자**로 쓰기)
  - String() -> 문자형으로 변환
  - Number() -> 숫자형으로 변환
    - true, false는 1, 0으로 변환
    - 문자는 NaN으로 변환
    - Number(null) -> 0, Number(undefined) -> NaN
  - Boolean() -> 불리언형으로 변환
    - false: 숫자 0, 빈 문자열 "", null, undefined, NaN
    - true: 문자 0, 공백

---

- alert: 알림
- prompt: 입력받음 -> 입력받은 값은 문자형


- 두 개의 값을 인수로 받을 수 있음 => 뒤의 값은 디폴트 값이 된다


- confirm: 확인받음
  - 확인과 취소 버튼 있음 => 확인이면 true, 취소면 false




- 단점: 스크립트 일시 정지, 스타링 X

---

a = 3 ->  a에 3을 넣어 주기

a == 3 -> a와 3을 비교, 값만 비교 (동등 연산자)

=== -> 타입까지 비교 (일치 연산자)

```
const isAdult = age > 19;
if(!isAdult) {
    console.log('미성년자는 출입 불가능합니다.')
}
```

- 이러한 방식으로 boolean 변수로 사용 가능

```
// 우선 순위
// 여자고, 이름이 Sally이거나 성인이면 통과

const gender = 'M';
const name = 'John';
const isAdult = true;

if(gender === 'F' && name === "Sally" || isAdult) {
    console.log('통과')
} else {
    console.log('실패')
}
// 이 경우 console에 '통과'로 찍힘
// AND 연산자가 우선순위가 높아서 'gender === 'F' && name === "sally"'가 먼저 처리되고,
// 그 이후에 isAdult가 처리되기 때문에

// 의도한 바대로 표현해 주려면 중간에 괄호를 쳐 줘야 한다
// 이름이 Sally이거나 성인
if(gender === 'F' && (name === "Sally" || isAdult)) {
    console.log('통과')
} else {
    console.log('실패')
}
```

---

## 반복문

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

- for ... in 반복문

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

- 반복문: for ... of

```
let Korean = ['가', '나', '다'];
for(let letter of Korean) {
    console.log(letter)
}
```

---

## 함수

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

sayHello(Tom);
```

=> "Hello, Tom" 출력

```
function sayHello(name) {
    let msg = 'Hello';
    if(name) {
    msg += `. ${name}`;
    }
    console.log(msg);
}
// 여기서 msg는 지역 변수여서 함수 밖에서는 못 씀
```

- 전역 변수보다는 함수에 특화된 지역 변수를 사용하는 것이 좋다

```
function sayHello(name) {
    let newName = name || `friend` // 매개 변수를 입력하지 않으면 friend 넣기
    let msg = `Hello, ${newName}`
    console.log(msg)
}
    sayHello(); // Hello, friend 출력
    sayHello('Jane'); // Hello, Jane 출력
```

```
// 매개 변수의 디폴트 값 설정하기
function sayHello(name = 'friend') {
    let msg = `Hello, ${name}`
    console.log(msg)
}
    sayHello(); // Hello, friend 출력
    sayHello('Jane'); // Hello, Jane 출력
```

- return으로 값 반환

```
fucntion add(num1, num2) {
    return num1 + num2;
}
const result = add(2,3);
console.log(result) // 5 출력
```

- return 할 게 없거나 return문만 있으면 undefined만 출력하고 종료
    - 함수를 종료하려는 목적으로 return만 사용하기도 함

```
function showError() {
  alert('에러가 발생했습니다.');
  return;
  alert('이 코드는 절대 실행되지 않습니다.')
}
const result = showError();
console.log(result);
```

⚓팁

- 한 번에 한 작업에 집중
- 읽기 쉽고 어떤 동작을 할 수 있는지 알기 쉽게 네이밍

- 함수 선언문: 어디서든 이용 가능
- 함수 표현식 : 이름 없는 함수 만들고 변수 선언해서 함수 할당, 코드에 도달해야만 생성

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

sayHello();
```

- 화살표 함수: function을 지우고 화살표 쓰기

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

// return문이 한 줄이라면 괄호 생략 가능
let add = (num1, num2) => num1 + num2;
```

```
// 인수가 하나면 괄호 생략 가능, 인수 없을 때는 괄호 생략 불가능
let sayHello = name => `Hello, ${name}`
```

---

## 객체

- 객체 생성

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

// 접근
superman.name // 'clark'
superman['age'] // 33

// 추가
superman.gender = 'male';
superman['hairColor'] = 'black';

// 삭제
delete superman.hairColor;
```

- 단축 프로퍼티

```
const name = 'clark';
const age = 33;

const superman = {
    name, // name: name
    age, // age: age
    gender: 'male';
}

// 프로퍼티 존재 여부 확인
superman.birthDay; // undefined
'birthDay' in superman; // false
'age' in superman; // true
```

```
function isAdult(user) {
    if(!('age') in user) || user.age < 20) { // 해당 조건을 넣어야 나이가 입력되지 않은 인물도
 거를 수 있음
    return false;
  }
    return true;
}

const Mike = {
```

```
    name: "Mike",
    age: 30
};

const Jane = {
    name: "Jane"
};

console.log(isAdult(jane))
```

```
// 객체 for ... in

const Mike = {
    name: "Mike",
    age: 30
};

for (x in Mike) {
  console.log(x)
} // name, age 출력

for(x in Mike) {
  console.log(Mike[x])
} // Mike, 30 출력
```

---

- method: 객체 프로퍼티로 할당된 함수

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

superman.fly(); // 날아갑니다. 출력

// function 생략 가능
 fly() {
  console.log('날아갑니다.')
 }
```

```
const user = {
    name: 'Mike',
    sayHello: function() {
        // 객체 명을 직접 쓰기보다는 this를 활용하는 것이 좋다
        console.log(`Hello, I'm ${this.name}`)
    }
```

```
}
    user.sayHello(); // Hello, I'm Mike 출력
```

---

- 배열: 순서가 있는 리스트

  -> 문자, 숫자, 객체, 함수 등을 표현할 수 있다

  .push(4) -> 배열 끝에 요소 추가

  .pop() -> 배열 끝에 요소 제거

  .unshift('가') -> 배열 앞에 요소 추가

  .shift() -> 배열 앞에 요소 제거