

# JAVASCRIPT

권정남

## ❖ 웹 개발자

### 풀스택(full stack) 개발자



사용자(user)



프론트 엔드(front-end) 개발자

- 웹 브라우저 화면에 보이는 모든 부분을 구성하는 작업

- HTML, CSS, **javascript**

JS가 !

정보 요청  
요청처리 응답



백 엔드(back-end) 개발자

- 화면에서 처리되는 작업과 데이터베이스 설계, 데이터 처리 등 웹사이트가 잘 운영되도록 하는 작업
- **JAVA**, PHP, 파이썬 등 프로그래밍 언어

## 01

# 자바스크립트 기초

- 자바스크립트 개요
- 변수와 상수
- 데이터 타입(자료형)
- 연산자
- 자료형 변환

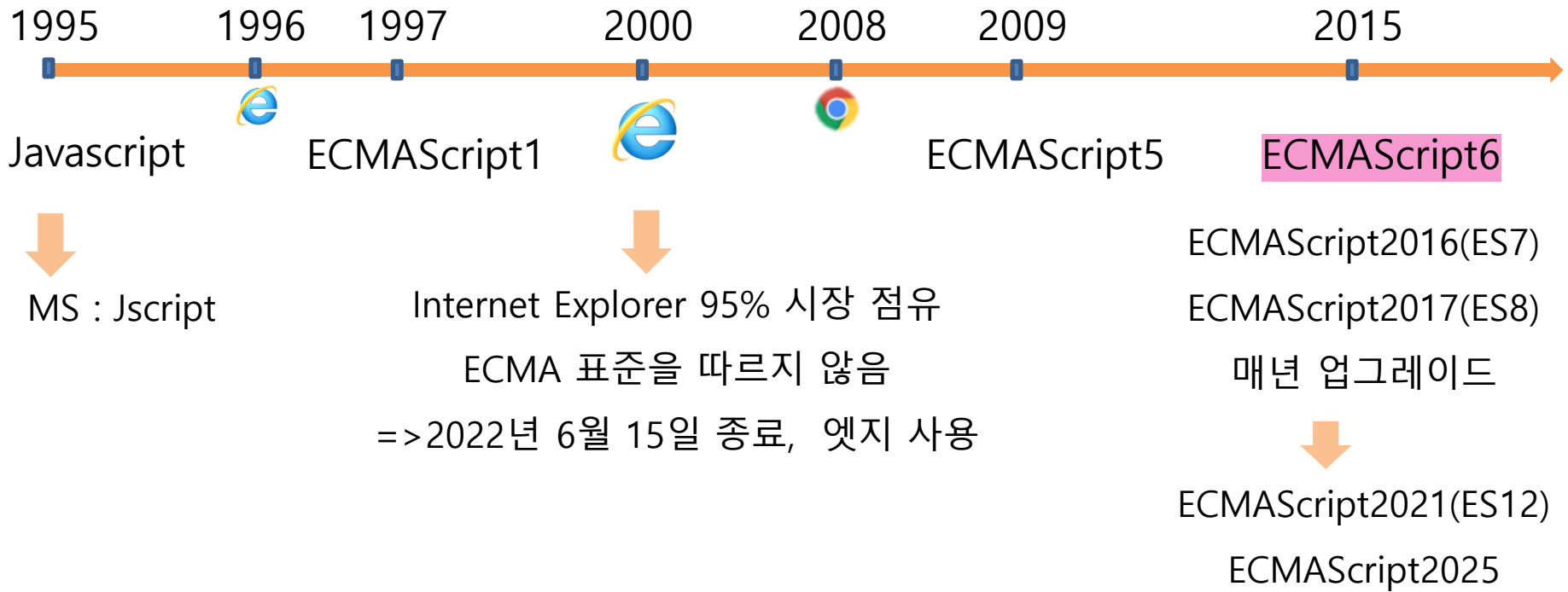
\*

가

!

## ❖ Javascript 역사

- 1993년 넷스케이프의 브랜드 아이크 '모카' 에서 시작
- 정적 웹 페이지 → 동적 웹 페이지 제작 → script 언어 추가
- Moca → Livescript → Javascript(1995년 : JAVA 언어의 인기에 얹힘)



## ❖ Javascript 버전

### 자바스크립트 역사

- 모카(넷스케이프 브랜드 아이크) → 라이브 스크립트 → 자바스크립트
- 유럽 컴퓨터 제조협회(ECMA)는 자바스크립트를 ECMAScript라는 이름으로 표준화(공식 명칭) – ECMAScript 1 : 1997년 6월 발표
- **ES2015=ES6** : 언어의 개념들이 대폭 변경되고 추가 되었음
- 최신 버전 : ES2025
- 2010년부터 웹 브라우저에서 벗어나 웹 서버(Node.js), 스마트폰 애플리케이션, 게임 개발, 데이터베이스 관리에 사용되는 등 활용범위가 가장 넓은 프로그래밍 언어가 되었고, 가장 인기 있는 언어가 되고 있음.

## ❖ javascript란 무엇일까요?

### Javascript는

- 웹 문서가 사용자의 동작에 반응할 수 있도록 해 줌으로써
- **웹을 동적으로 만들어 주고,**
- 웹에서 동작하는 프로그램을 만들며,
- 서버를 구성하고, **서버용 프로그램**을 만들 수 있는 프로그래밍 언어다.

### 특징

- 모든 웹 브라우저에서 작동,
- 웹 브라우저에서 실행 결과를 즉시 확인 할 수 있고
- 다양한 공개 API 사용할 수 있으며,
- 다양한 라이브러리와 프레임워크를 사용할 수 있다.

## ❖ 용어 정리

### 프로토타입 기반 : 객체를 원형(프로토타입)으로 동작하는 방식

- 객체 = 데이터 + 관련된 동작
- 객체지향 프로그래밍의 한 형태
- 객체의 동작 방식을 사용
- 프로토타입 지향, 인스턴스 기반 이라고도 한다.

### 스크립트 언어 : 동적인 웹 문서를 만들기 위한 웹 프로그래밍 언어

- 사용자 요청 → 처리 결과 → 사용자
- 독자적으로 실행되지 않고 다른 프로그램에 내장되어 사용
- 클라이언트 스크립트 언어 : javascript
- 서버 스크립트 언어 : ASP, PHP, JSP, Perl, CGI 등

## ❖ 용어 정리

### API : Application Programing Interface

- 어떤 정보(데이터)를 다른 사람이 쉽게 가져가 사용할 수 있도록 미리 만들어 둔 프로그램(일종의 모듈)
- 날씨정보 : 기상청에서 만든 자바스크립트 API 이용

### 라이브러리 : 자바스크립트로 미리 만들어 놓은 프로그램 모음

- jQuery, dojo, mootools

### 프레임워크 : 웹 프로그램을 쉽게 만들 수 있게 해 주는 틀

- 리액트(React), 앵귤러(Angular), 뷰(Vue) 등



## ❖ 자바스크립트 개발 환경

크롬 설치

비주얼 스튜디오 코드 설치

- HTML 문서 안에 자바스크립트 소스 작성
- 외부 스크립트 파일 연결

`<script src="js/script.js"></script>` script tag    src = "    "

- ✓ ■ 브라우저의 콘솔 도구 이용하기
  - ✓ 콘솔 도구에서 오류 확인

## ❖ 자바스크립트 기초

### 자바스크립트 선언문 html -> css -> javascript

- script 영역 선언 : <head>, <body> 모두 가능

<script>

실행문

</script>

스크립트가 있는 위치에서 실행되지만,

- 내부 자바스크립트는  </body> 앞에 작성
- 외부 자바스크립트는 <head> 영역에 작성

- 외부 자바스크립트 : 관리 용이


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

### 코드 작성 시 주의할 점

- 대소문자 구분, 한 줄에 한 문장 작성 권장 – 가독성 높임
- " ", ' ' 겹침 주의, 실행문 ( ) { } 짝이 맞도록 작성


## ❖ 자바스크립트 기초

### 웹브라우저에서 스크립트 해석 과정(p493)

- 
- ① **HTML 분석기** : 태그 분석
    - 웹 문서에 어떤 태그가 있는지, 관계는 어떻게 되는지 확인
  - ② **CSS 분석기** : 스타일 정보 분석
  - ③ **자바스크립트 해석기** : 스크립트 소스 해석
  - ④ ② ③의 분석 정보에 따라 화면에 표시
  - ⑤ 사용자 이벤트에 따라 자바스크립트 실행

## ❖ 자바스크립트 기초

### 표현식과 문장

- **표현식** : 값을 만들어 내는 간단한 코드
  - 288, 10+20
- **문장** : 하나 이상의 표현식이 모여 구성, 코드 단위(명령)
  - 10+20; (  => 문장 종결 부호)
  - 273;
  - let a = 30+40;
- **문장이 모여 프로그램 구성**
  - \* 문장 끝에 ; 입력하지 않아도 프로그램 실행에 문제 없지만 관례적으로 입력하는 것을 권장

## ❖ 출력문

### alert( '문자열' )

- 가장 기본 출력 방법
- 웹 브라우저에 **알림 창**으로 출력  
> alert('hello javascript')

### document.write( '문자열' )

- **웹 문서**에 문자열 출력  
> document.write('hello javascript')

### console.log( '문자열' )

- 개발자 도구 **console 창**에 출력( console 창: 소스실행, 오류 확인)  
> console.log('hello javascript')

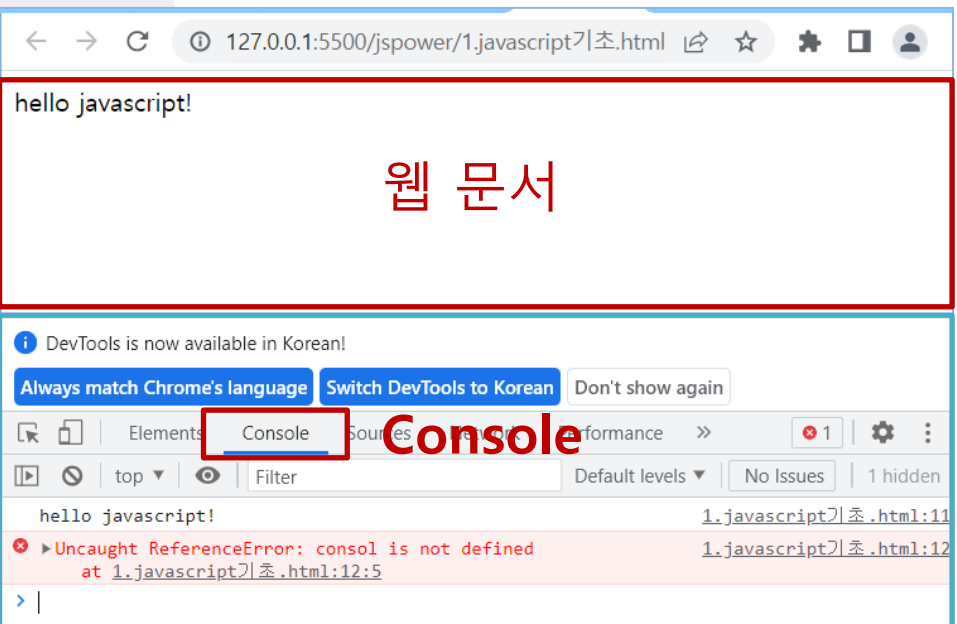
## ❖ 출력문

```
<body>
<script>
  document.write('hello javascript!');
  console.log('hello javascript!')
  alert('hello javascript!')
</script>
</body>
```

127.0.0.1:5500 내용:  
hello javascript!

alert 창

확인



## ❖ 입력문

### prompt( '문자열' , '기본값')

- 사용자로부터 **입력 값 받을 때** 사용
- 문자열 입력(숫자도 문자로 인식)
  - > prompt ('이름을 입력하세요')
  - > prompt ('이름을 입력하세요', '홍길동')

### confirm( '문자열' )

- 내용에 대한 확인(true/false)
- 확인 → true, 취소 → false 전달
  - > confirm('수락하시겠습니까?')

## ❖ 입력문

```
let input = prompt('숫자를 입력해 주세요', 5)  
let check = confirm('입력 값이 숫자 맞습니까?')  
console.log('입력값 : ' + input + ', 숫자 : ' + check + '입니다!')
```

127.0.0.1:5500 내용:  
숫자를 입력해 주세요

5

확인 취소

127.0.0.1:5500 내용:  
숫자를 입력해 주세요

100

확인 취소

127.0.0.1:5500 내용:  
입력 값이 숫자 맞습니까?

확인 취소

Elements Console Sources

top Filter

입력값 : 100, 숫자 : true입니다!



## ❖ 자바스크립트 기초

### 자바스크립트 코딩 규칙

- ① 코드를 보기 좋게 들여쓰기
  - 소스 간의 포함 관계를 알아 보기 쉽게 작성
- ② **세미콜론(;)으로 문장 구분(사용 권장)**
- ③ **공백을 넣어 코드를 읽기 쉽게 작성**
  - 예약어, 연산자, 값 사이 공백 - 가독성 높임
- ☆ ④ **주석 작성**
  - ✓ **한 줄 주석** **//** 가
  - **여러 줄 주석** : **/\* \*/**
- ⑤ **식별자**는 정해진 규칙에 맞게 작성

## ❖ 자바스크립트 기초

### 키워드(예약어)

- 언어가 만들어질 때 정해진 의미가 있는 단어
  - if, this, case, else, import 등



- 식별자로 사용할 수 없음

### 주석

- 프로그램 코드 설명문
  - HTML : `<!-- 설명문 -->`
  - javascript : 한 줄 주석 `//`

여러 줄 주석 `/* 설명문 */`

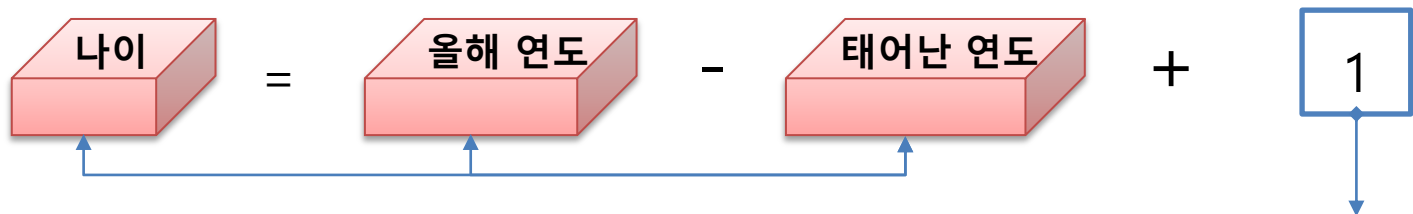
## ❖ 자바스크립트 기초(p508)

변수 선언 : **let** width **=** 100;

- 변수(Variable) : 프로그램에서 값을 담아두는 공간
- 변수 선언 : 값을 저장할 메모리 공간에 이름을 붙이는 것
- 변수 이름 : 어떤 값인지 추측할 수 있도록 의미 있게 작성

상수 선언 : **const** PI = 3.14

- 선언 이후 값 변경 불가, 선언할 때 값을 반드시 할당
- 재 선언 시 오류 발생



변수(Variable) : 변할 수 있는 값 (저장공간)

상수(Constant) : 변하지 않는 값

## ❖ 자바스크립트 기초

### 식별자 작성 규칙

- 이름을 붙일 때 사용하는 단어
  - 함수명, 변수명, 객체명 등
- 식별자 규칙
 

<ul style="list-style-type: none"> <li>- 키워드 사용 X</li> <li>- 숫자로 시작 X</li> <li>- 공백 포함 X</li> </ul>	<ul style="list-style-type: none"> <li>- 대소문자 구별</li> <li>- 특수문자 <u>\$</u> 허용</li> </ul> <p>(\$ → jQuery 식별자로 사용)</p>
---	---
- 개발자 사이의 관례
  - 한글 사용 가능하나 알파벳 사용
  - 의미 있는 단어 사용
  - 여러 단어인 경우 첫 글자는 대문자 (낙타표기법: birthYear)

jQuery

## ❖ 식별자 작성법

### ✓ 카멜케이스(camelCase)

- 두 번째 단어부터 대문자로 시작해 구분(method 선언)
- numberOfCollege

### 파스칼케이스(PascalCase)

- 모든 단어를 대문자로 시작해 구분(class 선언)
- NumberOfCollege

### 스네이크케이스(snake\_case)

- 언더스코어( \_ ) 넣어서 구분
- number\_of\_college


## ❖ 자바스크립트 기초

개발자 사이의 관례적 규칙 : 더 명확하게 식별자 의미 파악

- 식별자 종류
  - 생성자 함수명은 대문자로 시작
  - 변수, 인스턴스, 함수, 메소드 이름은 소문자로 시작
- 식별자 뒤에 괄호 있으면 → 함수(단독), 메소드(다른 식별자 O)
- 괄호 없으면 → 변수(단독), 속성(다른 식별자 O)
  - `alert('hi')` : 함수 가
  - ← - `Array.length` : 속성 . ( ) =>
  - `inputData` : 변수
  - `Math.abs(-273)` : 메소드

## ❖ 자바스크립트 자료형: 컴퓨터가 처리할 수 있는 자료의 형태

### 기본 자료형(data type)

- **문자열(string)** : 다음표로 묶인 모든 문자
  - "123", "안녕", '하세요', "this is \"string\"", "this is 'string'"
  - "안녕 \n하세요"  tab
  - 이스케이프 문자 : \t, \', \\*, \\
- **숫자(number)** : 다음표 없는 숫자, 사칙연산 가능
  - 123, 52.9
- **논리형(boolean)** : true(1), false(0) 값을 가짐
- **null** : 값이 유효하지 않거나, 값이 없다는 것을 명시
  - > let a=null; → typeof(a) : object 로 표시
- **undefined** : 변수가 선언되고 값이 할당 되지 않았을 때 유형
  - > let b;

## ❖ 자바스크립트 자료형

### 복합 자료형(data type)

- **array(배열)** : 하나의 변수에 여러 값을 저장하는 유형  
 > let seasons = [ '봄', '여름', '가을', '겨울'];
- **object(객체)** : 키와 값을 가진 데이터(속성)와 동작(메소드)을 함께 포함하는 유형  
 > let person = {  
     name : "kim",  
     age : 35;  
     eat : function(food) {}  
 };  

} --- 속성(property)  
 {} --- 메소드(method)



## ❖ 자바스크립트 자료형

### 자료형 검사 : typeof 연산자

- 변수에 저장하는 값에 따라 자료형 결정

> a="123"

> typeof a --- 'string'

> a=657

> typeof a --- 'number'

## ❖ 자료형 변환

### 자동 자료형 변환 : 숫자와 문자열

- 더하기(+) 연산 : 숫자를 문자열로 변환
- 나머지 연산들 : 문자열을 숫자로 변환

### 강제 자료형 변환 : 숫자와 문자열

- 숫자 자료형 : `Number('문자열')`  
숫자가 아닌 경우 NaN (Not a Number)  
\* `Number(null), Number(""), Number(' ') => 0`
- 숫자 정수 : `parseInt('문자열')`
- 문자열 자료형 : `String('숫자')`

## ❖ 자료형 변환

### 불 자료형 변환

- false 변환 : 0, NaN, "", null, undefined, 나머지 모든 데이터 true
- 강제 변환 : Boolean( data )



### 비교 연산자

- == : 자동으로 자료형이 변환되어 비교( != )
- === : 자료형과 값을 모두 비교( !== ) → 일치 연산자  
=== : type .

### 템플릿 문자열

- ` : 백쿼트(back quote) 기호로 문자열을 만들고,  
내부에 `${}` 사용하여 표현식이 계산되도록 하는 문자열

## ❖ 자료형 변환

함수	설명
Number()	문자열이나 논리형 -> 숫자 * <code>Number(null), Number(""), Number(' ') =&gt; 0</code>
parseInt()	문자열 -> 정수 숫자
parseFloat()	문자열 -> 실수 숫자
String()	숫자나 논리형 -> 문자열
Boolean()	(값) -> 논리형 * <code>false</code> 변환 : <code>0, NaN, "", "", null, undefined,</code> 나머지 모든 데이터 <code>true</code>

## ❖ 자료형 변환

```
let num = Number(input);
let str = String(input)
console.log('----- 더하기-----')
console.log('input : '+str + ', '+ (input + str))
console.log('input : '+num + ', '+ (input + num))
console.log('input : '+num + ', '+ (num + num))
```

```
console.log('----- 빼기-----')
console.log('input : '+str + ', '+ (input - str))
console.log('input : '+num + ', '+ (input - num))
console.log('input : '+num + ', '+ (num - num))
```

입력값 : 58, 숫자 : true입니다!

----- 더하기-----

input : 58, 5858

input : 58, 5858

input : 58, 116

----- 빼기-----

input : 58, 0

input : 58, 0

input : 58, 0

입력값 : abc, 숫자 : false입니다!

----- 더하기-----

input : abc, abcabc

input : NaN, abcNaN

input : NaN, NaN

----- 빼기-----

input : abc, NaN

input : NaN, NaN

input : NaN, NaN

## ❖ 자료형 변환

```
console.log('숫자 0의 불값은 ' + Boolean(0) + '입니다')
```

```
console.log(`NaN의 불값은 ${Boolean(NaN)} 입니다`)
```

```
console.log(Boolean(""))
```

```
console.log(Boolean(null))
```

```
console.log(Boolean('undefined'))
```

```
console.log(Boolean(undefined))
```

```
console.log(Boolean('false'))
```

```
console.log(Boolean(false))
```

```
console.log(Boolean(8))
```

숫자 0의 불값은 false입니다

NaN의 불값은 false 입니다

false

false

true

false

true

false

true

## ❖ 연산자

분류	기호	설명
사칙	<b>+</b> , -, *, /	연산순서 : () → [*, /] → [+ -]
연결	<b>+</b>	둘 이상의 문자열을 하나의 문자열로 만듦
제곱	**	$2^{**}3 \rightarrow 8$
나머지	%	$5 \% 3 \rightarrow 2$ <span style="color: red;">type</span>
증감	++, --	변수의 값을 1씩 증가, 감소
할당	=, +=, -=, *=, /=, %=	
 <b>비교</b>	>, <b>&gt;=</b> , <, <b>&lt;=</b> , <b>==</b> , <b>===</b> , !=, !== : 불 자료형에 활용	
<b>논리</b>	<b>&amp;&amp;(AND),   (OR), !(NOT)</b> : <u>불 자료형에 활용</u>	

가

## ❖ 연산 순서

( ) !

분류	연산 순서						
다른 분류	( ) → 단항 → 산술 → 비교 → 논리 → 할당 연산자						
단항	!	++	--				
산술	[**]	*	/	%	+	-	
비교	<	<=	>	>=	==	!=	=== !==
논리	&&						
할당	=	+=	-=	*=	/=	%=	



## ❖ 도전! 문제

### 문제1

가로와 세로 값을 입력 받아 사각형의 넓이를 계산하는 프로그램을 만들고, 결과값은 알림창으로 출력하세요.

prompt

alert

### 문제2

태어난 연도를 입력 받아 나이를 계산하는 프로그램을 만들고, 결과를 '태어난 연도는 0000년이고, 나이는 00살입니다'를 웹 문서에 출력하세요.

document

## ❖ 도전! 문제

**문제3** 결과 값을 예상해 보고 console 창에서 확인하세요

```
console.log(123 == '123')
```

```
console.log(123 === '123')
```

```
console.log(false != 'false')
```

```
console.log(false !== 'false')
```

```
console.log(typeof(256))
```

```
console.log(typeof('256'))
```

```
console.log(typeof(false))
```

```
console.log('10 + 20') → 10+20
```

```
console.log('10' + 20)
```

```
console.log('10' + '20')
```

```
console.log(10 + 20) → 30
```

```
console.log('10' * '2')
```

```
console.log('10' - '20')
```

```
console.log('10' % '3')
```



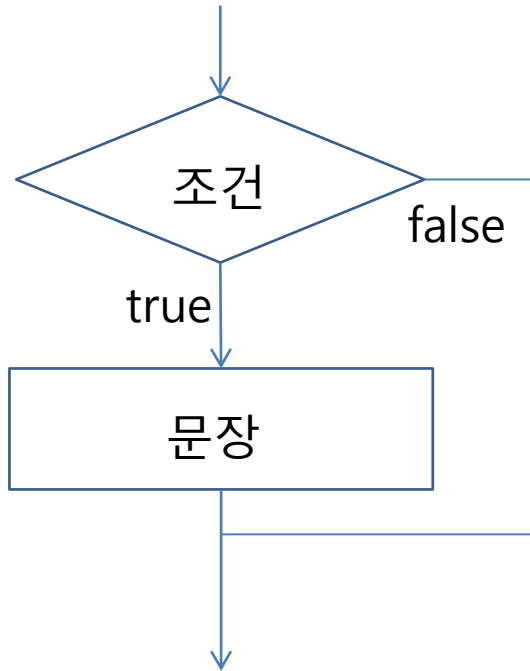
02

## 조건문

Very Important !!!!!

- If 조건문
- swich 조건문
- 삼항 연산자
- 짧은 조건문

## ❖ If 조건문 - 프로그램에서 특정 조건과 명령에 따라 실행 순서를 결정하는 문장



```

if(조건) {
    true 실행 문장
}
  
```

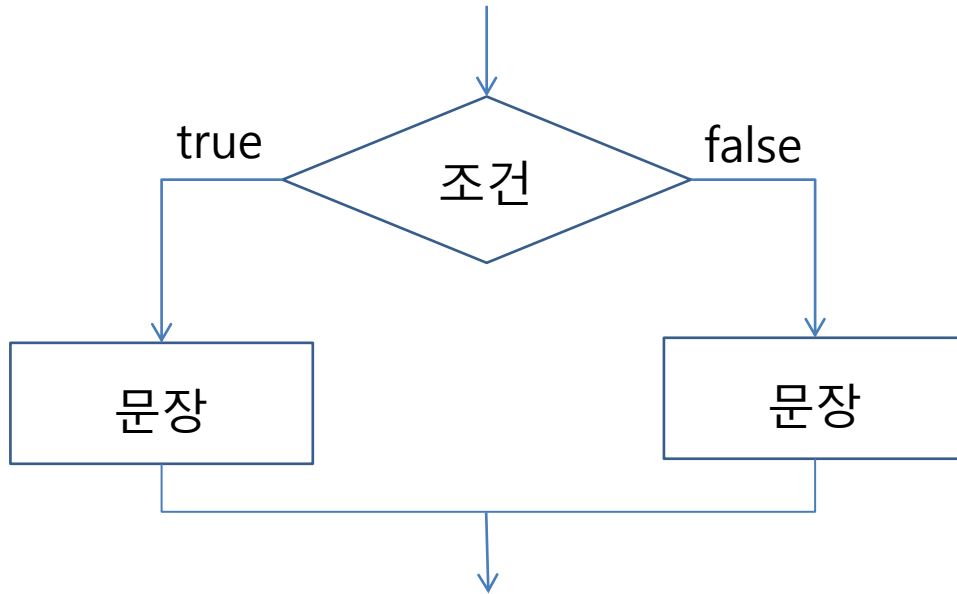
```

let a=10;
if(a>0){
    alert(`${a} 는 양수입니다`);
}
alert('end')
  
```

X

\* 실행되는 문장이 한 줄인 경우 {}생략 가능

## ❖ If else 조건문



```
If(조건) {  
    true 실행 문장  
}else{  
    false 실행 문장  
}
```

```
let a=10;  
if(a>0){  
    alert(`${a} 는 양수입니다`);  
}else{  
    alert(`${a} 는 양수가 아닙니다`);  
}
```

```
let a=10;  
if(a>0) alert(`${a} 는 양수입니다`);  
else alert(`${a} 는 양수가 아닙니다`);
```

## ❖ 중첩 if 조건문

```
if(조건) {  
    if(조건){  
        true 문장  
    }else{  
        false 문장  
    }  
}  
else{  
    false 문장  
}
```

```
let a=10;  
if(a >= 0){  
    if( a == 0){  
        alert(`a 는 zero입니다`);  
    }else{  
        alert(`${a} 는 양수입니다`);  
    }  
}  
else{  
    alert(`${a} 는 음수입니다`);  
}
```

## ❖ If else if 조건문

```
If(조건) {  
    true 문장  
}else if(조건){  
    true 문장  
}else{  
    false 문장  
}
```

```
let a=0;  
if(a > 0){  
    alert(`a 는 양수입니다`);  
}else if( a < 0){  
    alert(`a 는 음수입니다`);  
}else{  
    alert(`a 는 zero입니다`);  
}
```

## ❖ 도전! 문제

### 문제1

사용자에게 수를 입력 받아 홀수와 짝수를 구분하는 프로그램을 작성 해 보세요.

- 입력이 취소되면 '입력 취소' 출력
- 값이 입력되었을 경우 홀짝 구별

### 문제2

사용자에게 국어, 영어, 수학 점수를 입력 받아 평균을 구하고, 수,우,미,양,가를 구분하는 프로그램을 작성 해 보세요.



## ❖ 삼항 연산자, 짧은 조건문

(조건) ? true 실행문 : false 실행문 ;

- 한 줄로 코드를 표시할 수 있을 때 사용

```
let a=Number(prompt('숫자를 입력하세요', '숫자'));  
(a % 2 == 0) ? alert('짝수') : alert('홀수');
```

짧은 조건문 : 논리 연산자의 특성을 조건문으로 사용

- 논리합 : false || false 일 때 실행할 문장
- 논리곱 : true && true 일 때 실행할 문장

```
let a=Number(prompt('숫자를 입력하세요', '숫자'));  
a % 2 == 0 || alert('홀수');    → 좌변이 false면 우변 실행  
a % 2 == 0 && alert('짝수');    → 좌변이 true면 우변 실행
```

## ❖ switch 조건문 – 조건이 많은 경우 사용

```
switch (비교값){
```

```
  case 값:
```

```
    문장;
```

```
    break;
```

```
  case 값:
```

```
    문장;
```

```
    break;
```

```
  default:
```

```
    문장;
```

```
    break;
```

```
}
```

```
let a=Number(prompt('숫자를 입력하세요', '숫자'))
```

```
switch( a % 2 ){
```

```
  case 0 :
```

```
    console.log('짝수입니다')
```

```
    ✓ break
```

```
  case 1 :
```

```
    console.log('홀수입니다')
```

```
    break
```

```
  default : ex) 0 1 !
```

```
    console.log('숫자가 아닙니다')
```

```
    break
```

```
}
```

## ❖ 도전! 문제

### 문제1

사용자에게 수를 입력 받아 7로 나눈 나머지 값에 따라 요일이 콘솔창에 출력 되는 프로그램을 작성하세요.

0	1	2	3	4	5	6
일요일	월요일	화요일	수요일	목요일	금요일	토요일

### 문제2

사용자의 나이를 입력 받아

18세 이상이면 "00세 판매 가능!"

18세 미만이면 "00세 판매 불가!" 를 알림창에 출력하는 프로그램을 작성하세요

## 03

## 반복문

- while 문
- do while 문
- For 문

## ❖ 반복문

### 반복문은 왜 필요할까?

- 어떤 동작을 여러 번 실행할 때 사용

```
let sum = 0;
```

```
sum += 1;  
sum += 2;  
sum += 3;  
sum += 4;  
sum += 5;
```



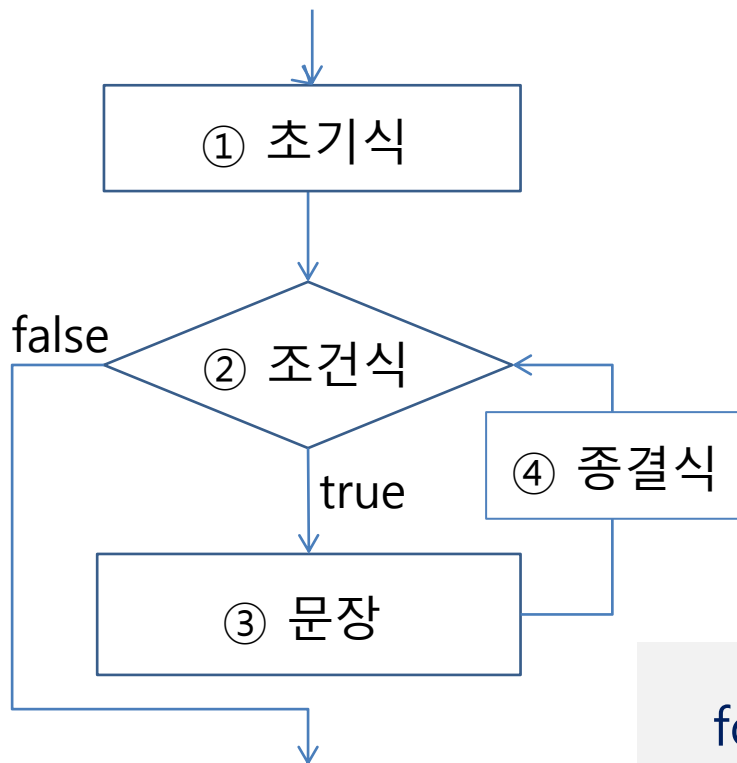
```
for(let i=1; i<=5; i++){  
    sum += i;  
}
```

가

```
console.log('1에서 5까지 합 : ' + sum);
```

## ❖ for문 : 조건보다 반복횟수에 비중을 둔 반복문

for



```
for( 초기값; 조건; 증감식 ){  
    반복할 문장  
}
```

```
for( let i = 1; i<=5 ; i++){  
    console.log(`${i}번째 출력`);  
}
```

## ❖ 중첩 반복문

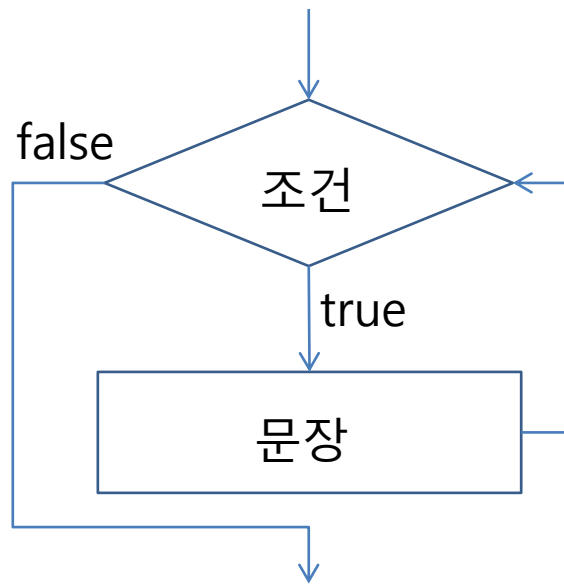
```

for(let i=1; i<=5; i++){
  let stars="";
  for( star=1; star<=i; star++){
    stars += "*";
  }
  console.log(stars)
}

```

행수	*개수	stars=*저장
i=1 → star=1회		*
i=2 → star=2회		**
i=3 → star=3회		***
i=4 → star=4회		****
i=5 → star=5회		*****

## ❖ while 문 : 조건을 비교해서 true인 동안 문장 실행

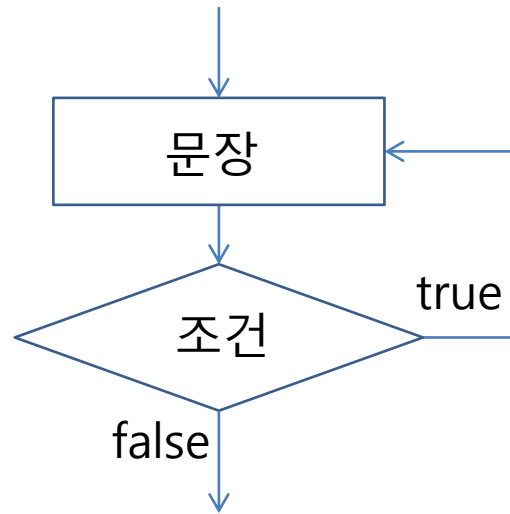


```
while (조건) {  
    반복할 문장  
}
```

```
let count = 0;  
while( count < 5 ){  
    console.log(`${count+1}번째 출력입니다`);  
    count ++;  
}
```



## ❖ do while 문 : 문장을 적어도 한번은 실행하고 조건을 비교



```
do {  
    반복할 문장  
} while (조건)
```

```
let count = 0;  
do {  
    console.log(`${count+1}번째 출력입니다`);  
    count ++;  
}while( count < 5 )
```

## ❖ 반복문

### ✓ 멈추는 **break**

- 조건문이나 반복문에서  
특정 조건에 만족하는 경우  
실행문을 멈추고 종료

### 건너뛰는 **continue**

- 특정 조건에 만족하는 경우,  
반복 작업을 멈추고,  
반복문의 처음으로 돌아가  
반복 작업을 진행

```
let sum = 0;
for(let i=1; i<=10; i++){
  if( i % 2 == 0) continue;
  sum += i;
  if( sum > 10) break;
}
console.log('합 : ' + sum);
```

## ❖ 도전! 문제

### 문제1

1에서 100까지 홀수 합과 짝수 합을 구하는 프로그램을 작성하세요

### 문제2

1에서 100까지 5의 배수의 합을 구하는 프로그램을 작성하세요

## ❖ 도전! 문제

**문제3** 숫자  $n$ 을 입력 받아서  $n!$ (팩토리얼)을 계산 하세요  
 $3! = 3*2*1=6$

**문제4** 2단 ~9단까지 구구단을 작성하는 프로그램을 작성하세요.

## ❖ 도전! 문제

**문제5** 숫자를 입력 받아 4의 배수인지 아닌지 출력하세요  
- 취소 버튼을 누르면 결과를 보여주지 않음

**문제6** 숫자를 입력 받아 1부터 입력 받은 수까지의 3의 배수 찾기  
- 3의 배수를 출력하고, 마지막에 3의 배수의 개수 출력

```
3,6,9,12,15,18,21,24,27,30,33,36,39,42,45,48
```

```
50까지의 3의 배수 개수 : 16
```

## ❖ 도전! 문제

## 문제7

입장객 수와 한 줄에 앉을 사람 수를 입력 받아 좌석을 배치하세요.

- 좌석 번호를 문서에 표로 만들기

좌석 1	좌석 2	좌석 3
좌석 4	좌석 5	좌석 6
좌석 7	좌석 8	좌석 9

숙제

## ❖ 도전! 문제

**문제8** 다음과 같이 출력되는 프로그램을 작성하세요

1

*
**
***
****
*****

2

*
**
***
****
*****