

Javascript 기초

JavaScript 개요 및 기본 문법

김태민

저번에 우리는

Bootstrap이란?

- Bootstrap은 반응형, 모바일 퍼스트 웹 개발을 위한 오픈소스 CSS 프레임워크
- HTML, CSS, JavaScript를 활용해 빠르고 일관된 UI 제공
- 2025년 기준, Bootstrap 5.0 사용 (jQuery 제거, 경량화)
- **모바일 퍼스트**: 기본 스타일이 작은 화면에 최적화, 미디어 쿼리로 큰 화면 지원
- **Flexbox 기반**: 유연한 레이아웃 구현
- **풍부한 컴포넌트**: 버튼, 폼, 네비게이션 바 등 사전 정의된 UI 제공
- **유틸리티 클래스**: 빠른 스타일링을 위한 클래스(text-center, d-flex 등)

빠른 프로토타이핑, 크로스 브라우저 호환성

Bootstrap Grid System

- Flexbox 기반의 12열 그리드 시스템. 컨테이너, 행, 열로 구성, 반응형 레이아웃 지원

구성 요소:

- 컨테이너: .container (고정 너비), .container-fluid (전체 너비)
- 행(Row): .row로 열 그룹화, 자동 간격(gutter) 제공
- 열(Column): .col-*로 12개 열 분할, 반응형 접두사(sm, md, lg, xl, xxl)

반응형 브레이크포인트:

- xs (<576px), sm (≥576px), md (≥768px), lg (≥992px), xl (≥1200px), xxl (≥1400px)

Components

- 웹 페이지의 사용자 인터페이스를 구성하는 재사용 가능한 구성 요소
- Bootstrap 5에서는 사전 스타일링된 컴포넌트를 제공하여 빠르고 일관된 UI 구현

Bootstrap 컴포넌트 사용 목적

- **사용자 경험(UX) 개선:** 직관적이고 접근 가능한 인터페이스 제공
- **개발 효율성:** 재사용 가능한 코드로 개발 시간 단축
- **일관성 유지:** 통일된 디자인 시스템으로 브랜드 정체성 강화

Components		
Accordion	Close button	Pagination
Alerts	Collapse	Popovers
Badge	Dropdowns	Progress
Breadcrumb	List group	Scrollspy
Buttons	Modal	Spinners
Button group	Navs & tabs	Toasts
Card	Navbar	Tooltips
Carousel	Offcanvas	

Utility Class

- 간단한 CSS 클래스들로, 빠르고 직관적인 스타일링과 레이아웃 조정을 가능하게 함
- 커스텀 CSS 작성 없이 요소의 정렬, 여백, 디스플레이, 색상 등을 조정
- 복잡한 CSS 규칙 대신 간단한 클래스 이름
- **반응형 디자인**: 반응형 접두사(sm, md, lg 등)를 활용해 다양한 화면 크기에 대응

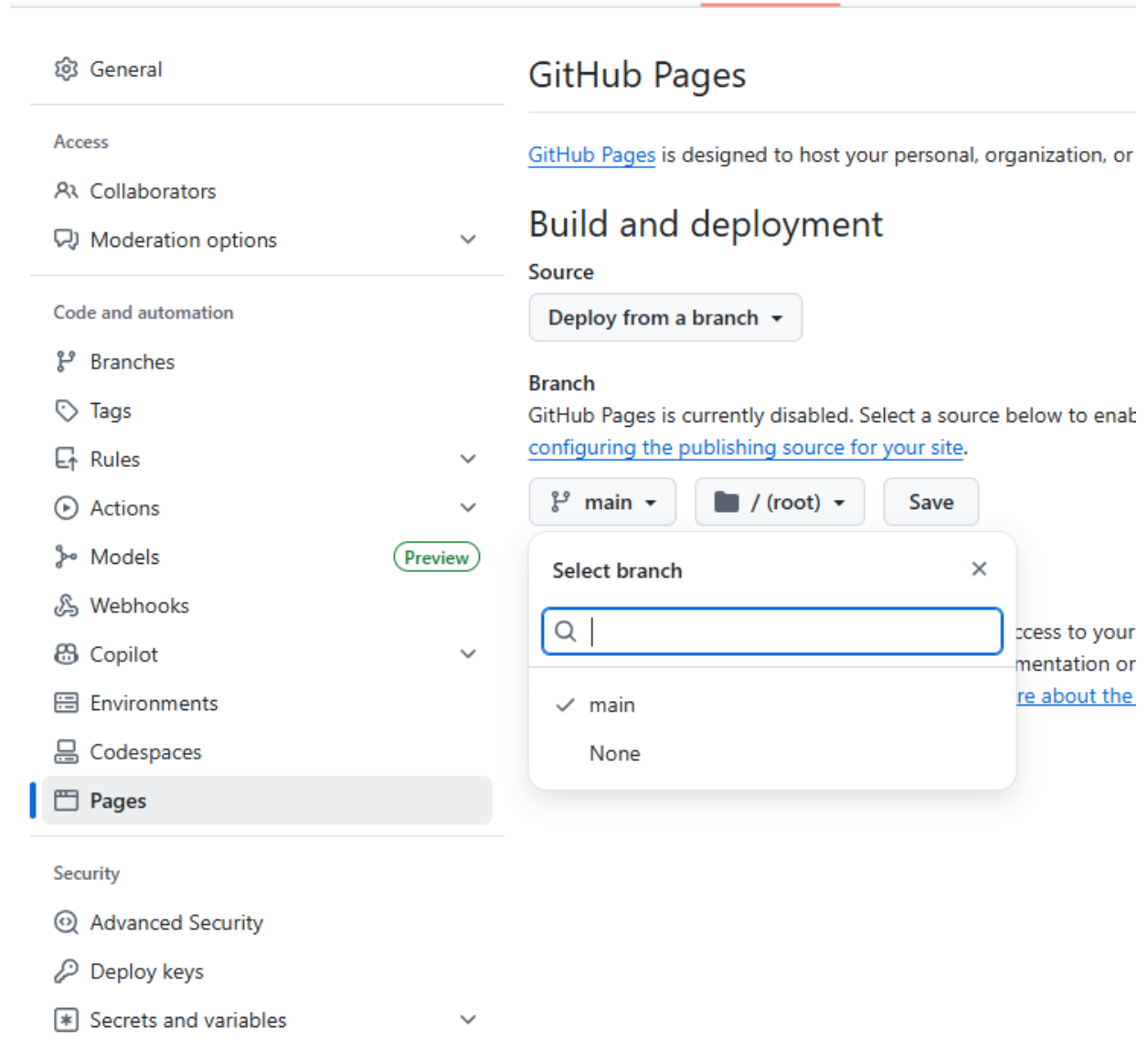
(*) Utilities

API

	Float	Shadows
Background	Interactions	Sizing
Borders	Link	Spacing
Colors	Object fit	Text
Display	Opacity	Vertical align
Flex	Overflow	Visibility
	Position	Z-index

새로운 git repository 생성

좌측 Pages 메뉴 선택
Deploy from a branch
Branch : main



The screenshot displays the GitHub repository settings interface. On the left sidebar, the 'Pages' menu item is highlighted. The main content area shows the 'GitHub Pages' settings. Under the 'Build and deployment' section, the 'Source' is set to 'Deploy from a branch'. The 'Branch' dropdown is set to 'main'. A 'Select branch' modal is open, showing a search bar and a list of branches with 'main' selected. The 'Save' button is visible next to the branch selection.

실습 환경

OS : Window / Mac

브라우저 : Chrome

에디터 : VS Code <https://code.visualstudio.com/>

VS Code 익스텐션 : ESLint, Prettier, HTML CSS Support, HTML to CSS autocompletion, Auto Rename Tag, Auto Close Tag, htmltagwrap

Git : git bash, github 가입

Javascript 개요

프로그래밍이란?

مرحبًا وابتهج اليوم!

프로그래밍이란?

مرحبًا وابتهج اليوم!

안녕하세요. 오늘 하루도 힘내요!

프로그래밍이란?

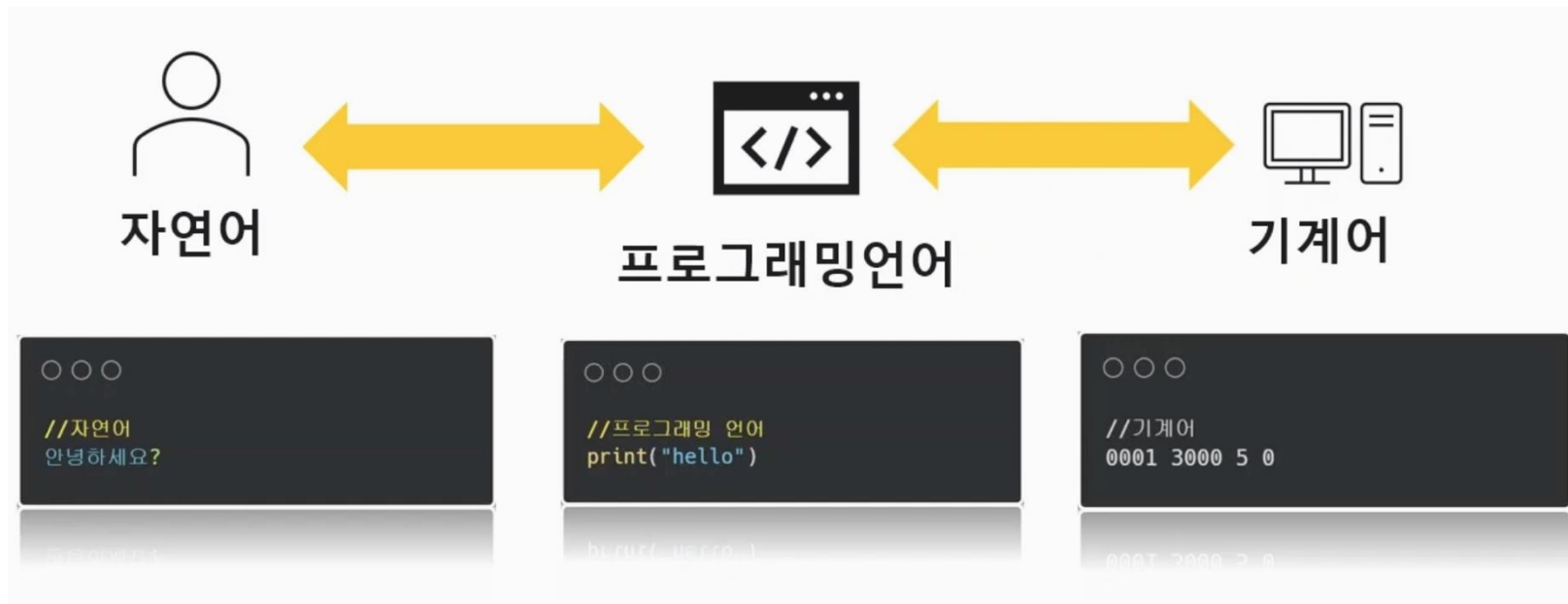
ARABIC ALPHABET

خ ح ج ث ت ب ا
ص ش س ز ر ذ د
ق ف غ ع ظ ط خ
ي و ه ن م ل ك

	ㅏ	ㅑ	ㅓ	ㅕ	ㅗ	ㅛ	ㅜ	ㅠ
ㅏ	가	거	고	구	그	기	개	게
ㅑ	카	커	코	쿠	크	키	캐	케
ㅓ	나	너	노	누	느	니	내	네
ㅕ	다	더	도	두	드	디	대	데
ㅗ	타	터	토	투	트	티	태	테
ㅛ	라	러	로	루	르	리	래	레
ㅜ	마	머	모	무	므	미	매	메
ㅠ	바	버	보	부	브	비	배	베
ㅡ	파	퍼	포	푸	프	피	패	페
ㅚ	사	서	소	수	스	시	새	세
ㅜ	자	저	조	주	즈	지	재	제
ㅡ	차	처	초	추	츠	치	채	체
ㅛ	아	어	오	우	으	이	애	에
ㅎ	하	허	호	후	흐	히	해	헤

프로그래밍

- 컴퓨터가 특정 작업을 수행하도록 지시하는 명령어 집합을 작성하는 과정
- 인간의 언어(한국어, 영어 등)로 컴퓨터와 소통하기 위한 도구
- 예: 웹사이트에서 버튼 클릭 시 팝업 표시, 스마트폰 앱에서 알림 전송



Javascript 개요

프로그래밍

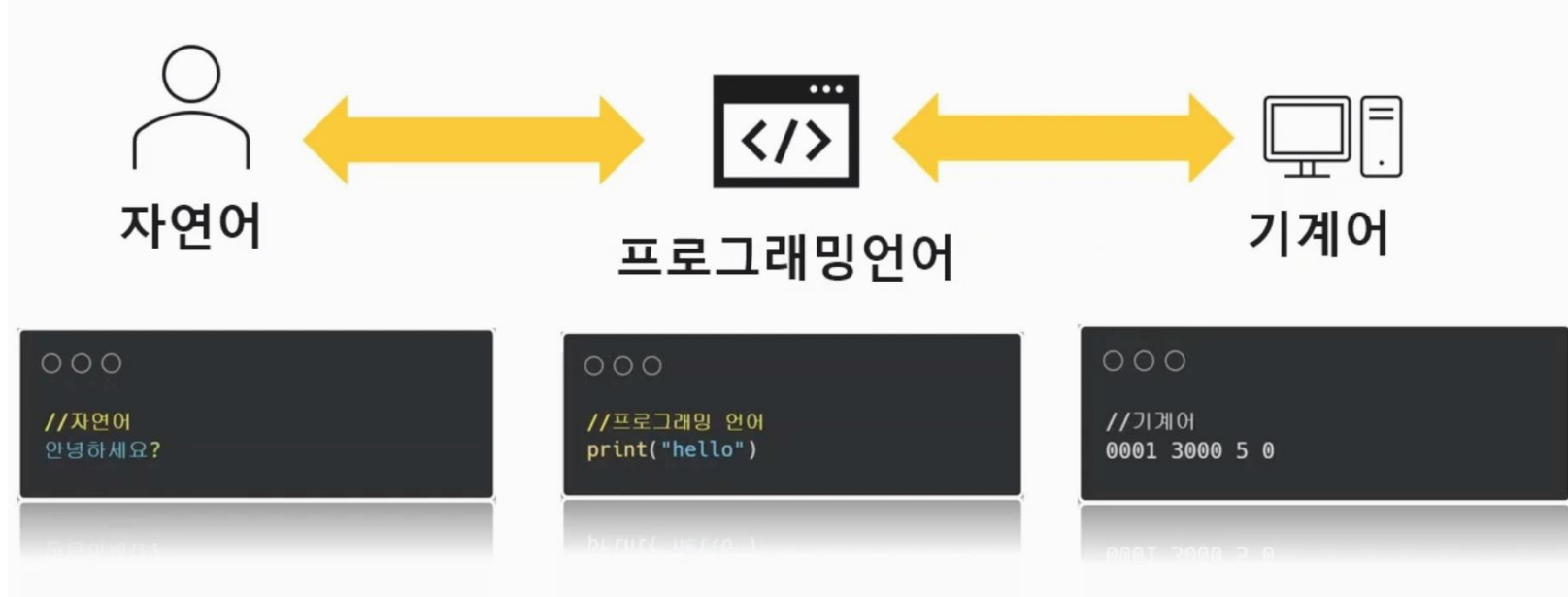
- 컴퓨터가 특정 작업을 수행하도록 지시하는 방법
- 인간의 언어(한국어, 영어, 중국어, ...)
- 예: 웹사이트에서 버튼 클릭 시 반응

기계를 번역하는 방식에 따라
컴파일드 언어(Compiled Language)와 인터프리티드 언어(Interpreted Language)

컴파일드 언어는 다수의 명령어로 이뤄진 소스코드를 한 번에 기계어로 번역해서 실행 파일을 생성. [C, C++, C#, GO, Java]

반면, 인터프리티드 언어는 소스코드를 한 줄씩 기계어로 번역해서 실행 결과를 보여줌. 이때문에 인터프리티드 언어는 스크립트(Script)언어라고도 한다.

[Javascript, Python, Ruby, PHP, shell script]



Javascript로 시작하는 프로그래밍

- 웹 개발의 핵심 언어, 브라우저에서 바로 실행
- 프론트엔드(React, Vue)와 백엔드(Node.js) 모두 가능
- 초보자 친화적: 간단한 코드로 즉각적인 결과 확인



배경:

1990년대 초: 웹은 HTML로 정적인 텍스트/이미지 페이지 중심
동적 기능(폼 검증, 버튼 상호작용) 필요성 대두

개발 과정:

1995년: Netscape의 Brendan Eich가 "Mocha개발, "LiveScript " "JavaScript"로
명명

1996년: Netscape Navigator 2.0에 JavaScript 탑재, 웹의 동적 기능 시작

1997년: ECMAScript 표준화(ECMA-262), JavaScript의 공식 표준

2000년대: AJAX로 비동기 데이터 로딩, jQuery로 쉬운 DOM 조작

2010년대: Node.js(2009)로 서버 개발, React/Vue/Angular로 현대적 웹 앱

2025년 현재: ES6+(2015~)로 최신 문법, 다양한 프레임워크/라이브러리

인터프리터 언어:

코드를 즉시 실행, 컴파일 불필요

동적 타이핑:

변수 타입을 선언 없이 실행 중 결정

(예: `let x = 5; x = "Hello";`)

객체 기반:

객체를 활용해 데이터와 동작 정의

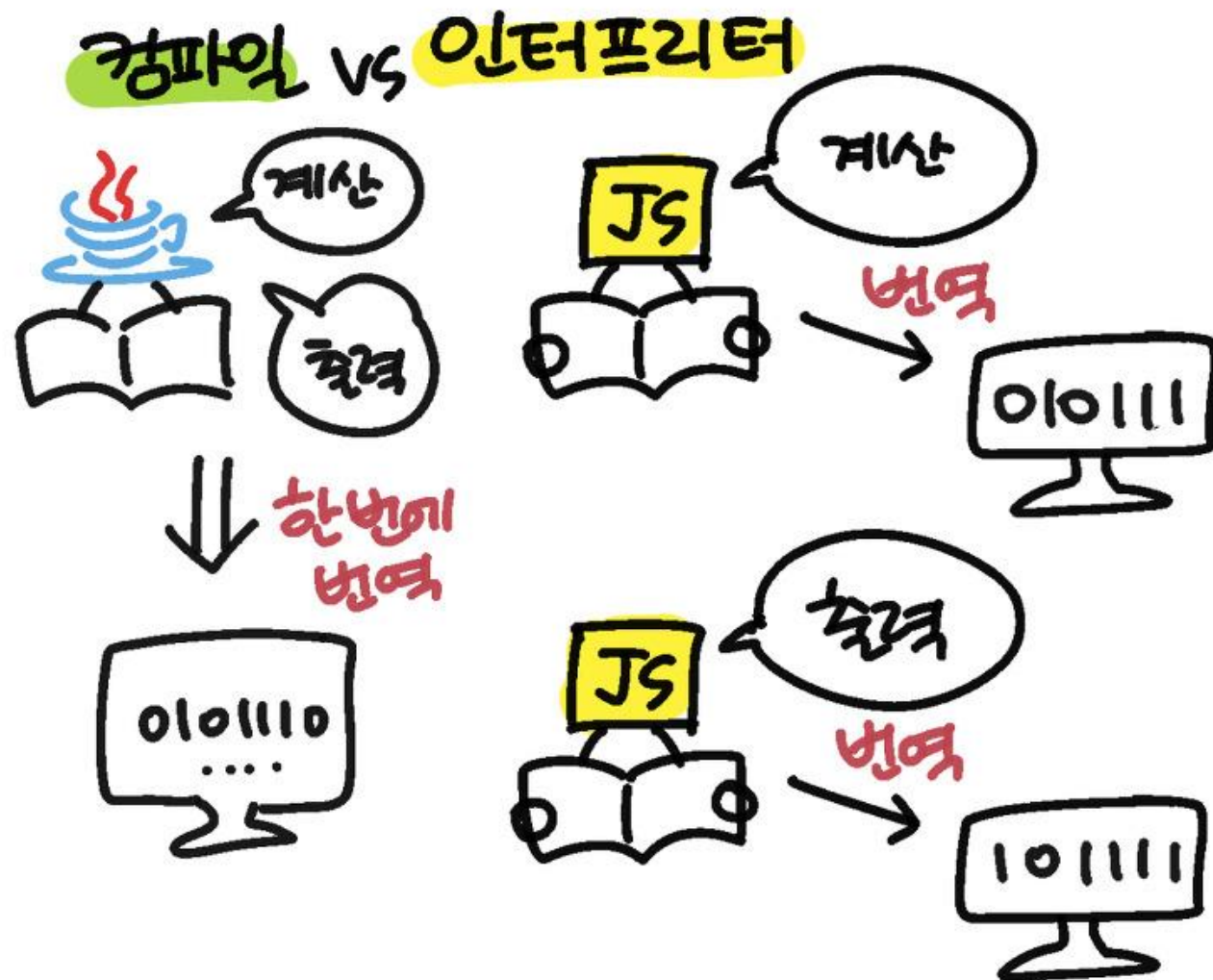
이벤트 중심:

사용자 행동(클릭, 스크롤, 입력)에 반

비동기 처리:

Promise, `async/await`로 서버 요청, 타이머 등 처

리



클라이언트 측 실행:

브라우저에서 즉각 실행, 빠른 사용자 경험

서버 측 지원:

Node.js로 백엔드 개발(예: API 서버, 데이터베이스 연동)

크로스 플랫폼:

웹, 모바일 앱(React Native), 데스크톱 앱(Electron)

호환성:

모든 현대 브라우저(Chrome, Firefox, Safari) 지원

풍부한 생태계:

React, Vue, Node.js 등으로 확장 가능

Client

PC Browser
Mobile Chrome App
Mobile App

NextJS

Server

RESTful Api
Oauth
RAG / LLM Api

NodeJS Express

DB

UserInfo
ProductInfo
Payment ...

MongoDB / MySQL

AWS / GCP / Azure / Cafe24 ...

Javascript 기본문법

키워드(keyword)

자바스크립트 프로그래밍 언어에서
어떤 역할이나 기능이 정해진 특별한
단어

다른 용어로 **예약어(reserved word)**

예를 들어, 변수라는 공간을 생성할
때는 var, let 키워드를 사용

async	await	break	case	catch
class	const	continue	debugger	default
delete	do	else	enum	export
extends	false	finally	for	function
if	implement	important	in	instanceof
interface	let	new	null	package
private	protected	public	return	static
super	switch	this	throw	try
ture	typeof	void	while	with
yield				

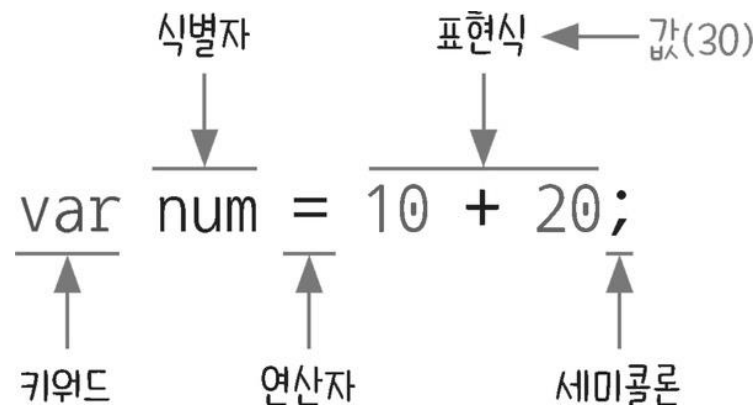
변수 선언, 할당, 초기화

변수를 생성하고 값을 저장하는 문법에서 var 키워드나 이후에 배우는 let, const 키워드를 사용
'변수의 식별자를 지정하는 행위를 **변수를 선언한다**고 부름.

할당 연산자인 = 기호로 우변에 있는 값을 변수 공간에 대입(저장)하는 것을 **값을 할당한다**고 함.

변수는 초기에 값을 할당하지 않고 선언만 할 수 있음.

```
var num;
```



변수 var

ES5 시절 사용

재선언/재할당 가능, 호이스팅(선언 전 사용 가능) 문제

단점: 스코프 혼란, 현대적 코드에서 사용 지양

```
var name = "홍길동";
```

```
var name = "김영희"; // 재선언 가능
```

```
name = "이철수"; // 재할당 가능
```


변수 let

ES6 도입

재선언 불가. 재할당 가능

장점: 명확한 스코프, 변수 변경 시 유용

```
let age = 25;
```

```
// let age = 30; // 오류: 재선언 불가
```

```
age = 30; // 재할당 가능
```

상수 const

ES6 도입, 블록 스코프 기반

재선언/재할당 불가, 상수(constant) 선언 시 사용

장점: 값 고정, 안정적 코드 작성

주의: 객체/배열의 내부 속성은 변경 가능

```
const pi = 3.14;
```

```
// pi = 3.14159; // 오류: 재할당 불가
```

```
const user = { name: "홍길동" };
```

```
user.name = "김영희"; // 객체 내부 속성 변경 가능
```

// 변수 선언 실습

var greeting = "안녕하세요"; // 오래된 방식

let username = "홍길동"; // 변경 가능한 변수

const maxAttempts = 3; // 고정된 상수

console.log(greeting); // "안녕하세요"

username = "김영희"; // 재할당

console.log(username); // "김영희"

console.log(maxAttempts); // 3

강제적 명명 규칙

변수나 상수를 선언하고 식별자를 지정할 때 몇 가지 규칙

이 규칙은 언어적 차원에서 강제적인 것도 있고, 관용적인 것도 있음.

강제 규칙은 지키지 않을 경우 프로그래밍 언어 자체에서 오류 발생

규칙	불가능 예
식별자에 키워드 사용 불가	var, let, const
식별자에 공백 포함 불가	my School, like food
식별자의 첫 글자는 영문 소문자, _(언더스코어), \$ 기호 만 사용	*name, #age, @email

관용적 명명 규칙

관용 규칙은 지키지 않아도 프로그래밍 언어에서 오류를 발생시키진 않지만, 되도록 지키는 것이 좋음.

규칙	좋은 예	나쁜 예
식별자는 영문으로만 작성	name, age	이름, 나이
식별자는 의미 있는 단어로 작성	name, age(이름과 나이 저장 시)	a, b(이름과 나이 저장 시)

식별자 표기법

자바스크립트에서 식별자를 표기하는 방법은 대표적으로 카멜 표기법(camel case), 언더스코어 표기법(underscore case), 파스칼 표기법(pascal case)

표기법	설명	예
카멜 표기법	변수명과 함수명 작성 시 사용	firstName, lastName
언더스코어 표기법	상수명 작성 시 사용	FIRST_NAME, last_name
파스칼 표기법	생성자 함수명 작성 시 사용	FirstName, LastName

자료형

자료형(data type)이란 자바스크립트에서 사용할 수 있는 데이터의 종류를 의미.

자바스크립트의 자료형은 **기본 자료형**과 **참조 자료형**으로 구분.

기본(primitive) 자료형으로는 문자(string), 숫자(number), 논리(boolean), undefined, null, Symbol
자료형이 있고, 참조(reference) 자료형에는 객체(object)가 있음.

```
console.log(typeof 변수명);
```

기본자료형 – 문자 string

큰따옴표("")나 작은따옴표(')로 둘러싸인 값의 형태

큰따옴표로 시작했으면 반드시 큰따옴표로 끝나야 하고, 작은따옴표로 시작했으면 작은따옴표로 끝나야 함

시작과 끝의 따옴표가 다르면 오류

```
let string1 = "Hello, World!";
```

```
let string2 = 'Hello, World!';
```

```
let string3 = "Hello, World!';
```

```
let string4 = 'Hello, World!";
```


기본자료형 – 문자 string

문자열에 따옴표가 포함된 경우,

```
let string1 = '문자열은 큰따옴표(")로 감싸면 됩니다.;
```

```
let string2 = "문자열은 작은따옴표(')로 감싸면 됩니다.;"
```

혹은 문자열 연결 연산자(+) 또는 이스케이프 문자로 해결

기본자료형 – 문자 string

문자열 연결 연산자(+)는 우리가 알고 있는 덧셈 기호

```
let string = '문자열' + " 연결 연산자";
```

```
let string = '문자열은 큰따옴표(")나' + " 작은따옴표(')로 감싸면 됩니다.";
```

기본자료형 – 문자 string

이스케이프 문자는 웹 브라우저가 사용자 의도와 다르게 문자열을 해석할 때 사용

let string = '문자열은 큰따옴표(")나 작은따옴표(① \')로 감싸면 됩니다.');

이스케이프 문자 사용 시	설명
\'	작은따옴표(single quotes)
\"	큰따옴표(double quotes)
\n	줄 바꿈(new line)
\t	수평 탭(horizontal tab)
\\	역슬래시(backslash)

기본자료형 – 문자 string

문자열 연결 연산자(+)는 우리가 알고 있는 덧셈 기호

```
let string = '문자열' + " 연결 연산자";
```

```
let string = '문자열은 큰따옴표(")나' + " 작은따옴표(')로 감싸면 됩니다.';
```

기본자료형 - 문자 string

ES6에서 추가된 템플릿 문자열은 백틱(`)으로 문자열을 정의하는 방법
작은따옴표, 큰따옴표를 사용하지 않으며 다양한 단점 극복

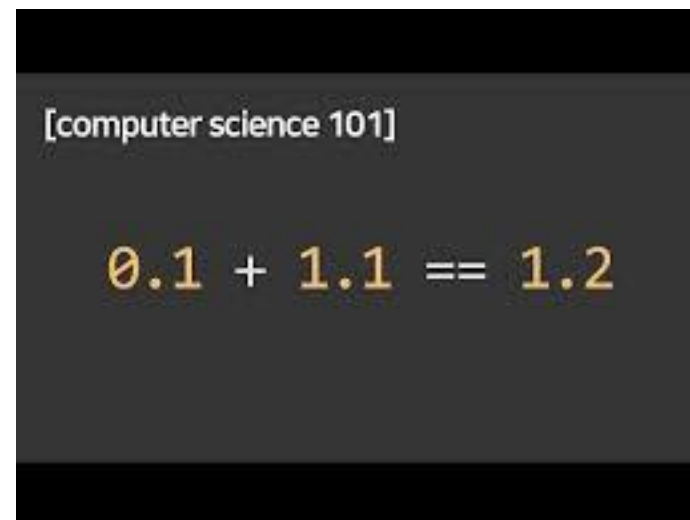
1. Enter를 눌렀을 때 줄 바꿈이 적용됩니다. 그래서 이스케이프 문자를 사용하지 않아도 됩니다.
2. `${}` 문법을 이용해 문자열에 변수 또는 식을 넣을 수 있습니다

기본자료형 - 숫자 number

자바스크립트는 정수와 실수를 구분하지 않고 전부 하나의 숫자 자료형(숫자형)으로 취급
10이나 0.1이나 자바스크립트는 같은 숫자형

실수 계산에서는 유의가 필요함

<https://www.youtube.com/watch?v=-GsrYvZoAdA>



기본자료형 - 논리형 boolean

참 또는 거짓에 해당하는 논리 값인 true, false를 의미

```
let boolean1 = true;
```

```
let boolean2 = false;
```

```
let boolean3 = 10 < 20;
```

```
let boolean4 = 10 > 20;
```

특수자료형 undefined

변수나 상수를 컴퓨터 메모리 공간에 선언하면 반드시 생성한 공간에 저장할 데이터를 할당해야 함.

할당하지 않을 경우 자바스크립트 내부적으로 변수와 상수 공간에 임시로 데이터를 할당
이때 할당되는 값이 undefined입니다.

```
let empty;
```


특수자료형 null

변수나 상수를 선언하고 의도적으로 선언한 공간을 비워 둘 때 할당

```
let empty = null;
```

Symbol

심볼(Symbol)은 유일한 값을 가진 변수 이름을 만들 때 사용
심볼에 설명을 붙여서 의미를 부여할 수도 있습니다.

심볼은 유일무이한 값을 가지기 때문에, 같은 설명을 붙인 심볼끼리도 비교하면 false가 반환됩니다.

```
const user = Symbol();  
const user = Symbol('this is a user');  
const symbolA = Symbol('this is Symbol');  
const symbolB = Symbol('this is Symbol');  
console.log(symbolA === symbolB); // false
```

BigInt

아주 큰 정수(Integer)를 표현하기 위해 등장한 데이터 타입
숫자 뒤에 n을 붙이거나 BigInt 함수를 사용

소수 표현 불가: BigInt는 큰 정수를 표현하기 위한 타입이기 때문에 소수 표현에는 사용할 수 없습니다.

정수로 결과 반환: 소수 형태의 결과가 리턴되는 연산은 소수점 아래 부분을 버리고 정수 형태로 반환합니다.

타입 간 연산 불가: BigInt 타입끼리만 연산할 수 있으며, 서로 다른 타입끼리의 연산은 명시적으로 타입 변환을 해야 합니다.

BigInt

```
console.log(9007199254740993n); // 9007199254740993n
```

```
console.log(BigInt('9007199254740993')); // 9007199254740993n
```

```
1.5n; // SyntaxError
```

```
console.log(10n / 6n); // 1n
```

```
console.log(5n / 2n); // 2n
```

```
console.log(3n * 2); // TypeError
```

```
console.log(3n * 2n); // 6n
```

```
console.log(Number(3n) * 2); // 6
```

객체

객체(object)는 자바스크립트의 핵심적인 자료형

앞에서 언급한 기본 자료형을 제외하고 자바스크립트에서 거의 모든 데이터와 자료구조는 객체
객체 자료형에서 파생되는 자료형으로 배열, 객체 리터럴, 함수 등이 있음

객체 - 배열

복수의 데이터를 정의할 수 있는 자료형

배열로 정의한 데이터를 **요소**. 배열 요소에 접근하려면 **인덱스(index)**를 이용.

인덱스는 배열에서 각 데이터가 있는 위치를 가리키는 숫자.

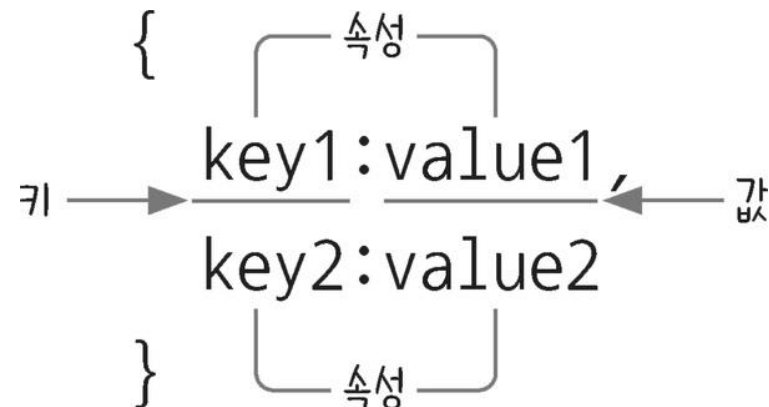
인덱스 숫자는 0부터 시작

```
let studentScore = [80, 70, 90, 60];  
let emptyArray = [];  
let array = ['abc', 10, true, undefined, null, [], {}, function(){}];
```

객체 - 객체 리터럴

객체를 정의할 때 중괄호({})를 사용

중괄호 안에는 **키**(key)와 **값**(value)의 한쌍으로 이루어진 **속성**(property)이 들어감



```
let studentScore = {  
  koreanScore:80,  
  englishScore:70,  
  mathScore:90,  
  scienceScore:60  
};  
console.log(studentScore.koreanScore); // 80  
console.log(studentScore['englishScore']); // 70
```

오늘 우리는

Javascript 개요

프로그래밍

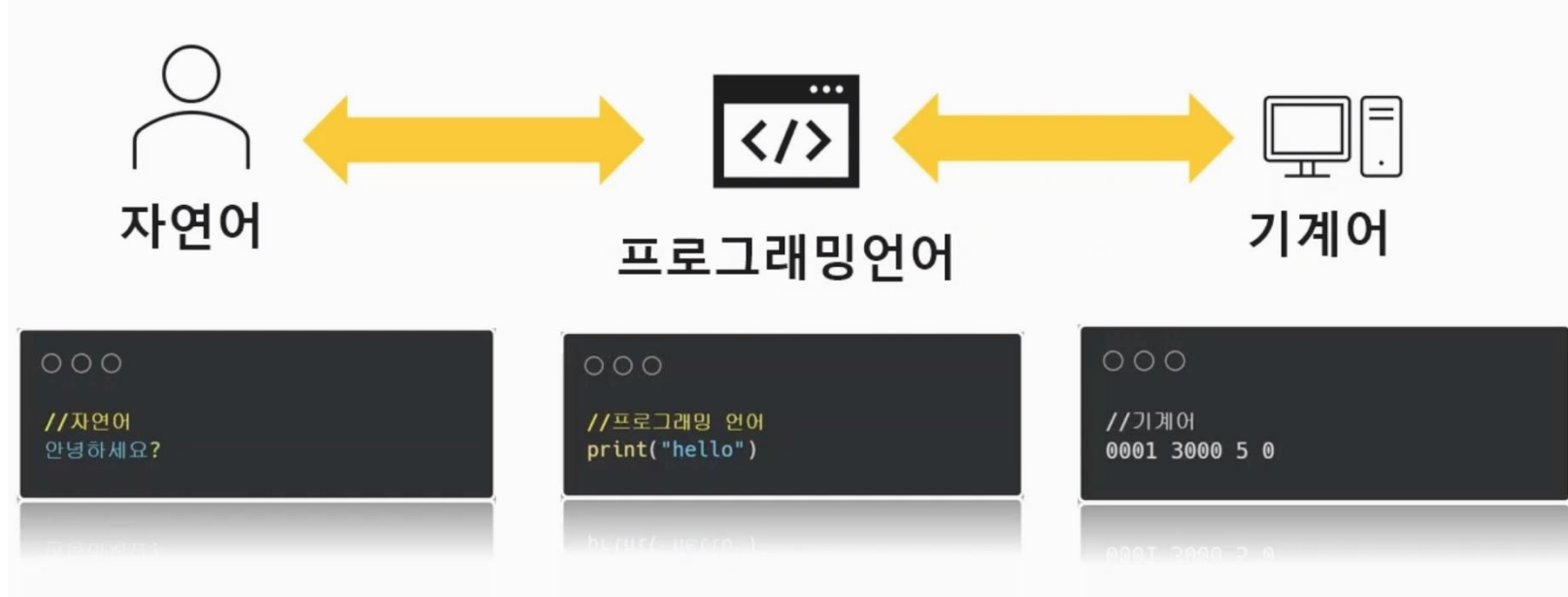
- 컴퓨터가 특정 작업을 수행하도록 지시하는 방법
- 인간의 언어(한국어, 영어, 중국어 등)
- 예: 웹사이트에서 버튼 클릭 시 발생하는 동작

기계를 번역하는 방식에 따라
컴파일드 언어(Compiled Language)와 인터프리티드 언어(Interpreted Language)

컴파일드 언어는 다수의 명령어로 이뤄진 소스코드를 한 번에 기계어로 번역해서 실행 파일을 생성. [C, C++, C#, GO, Java]

반면, 인터프리티드 언어는 소스코드를 한 줄씩 기계어로 번역해서 실행 결과를 보여줌. 이때문에 인터프리티드 언어는 스크립트(Script)언어라고도 한다.

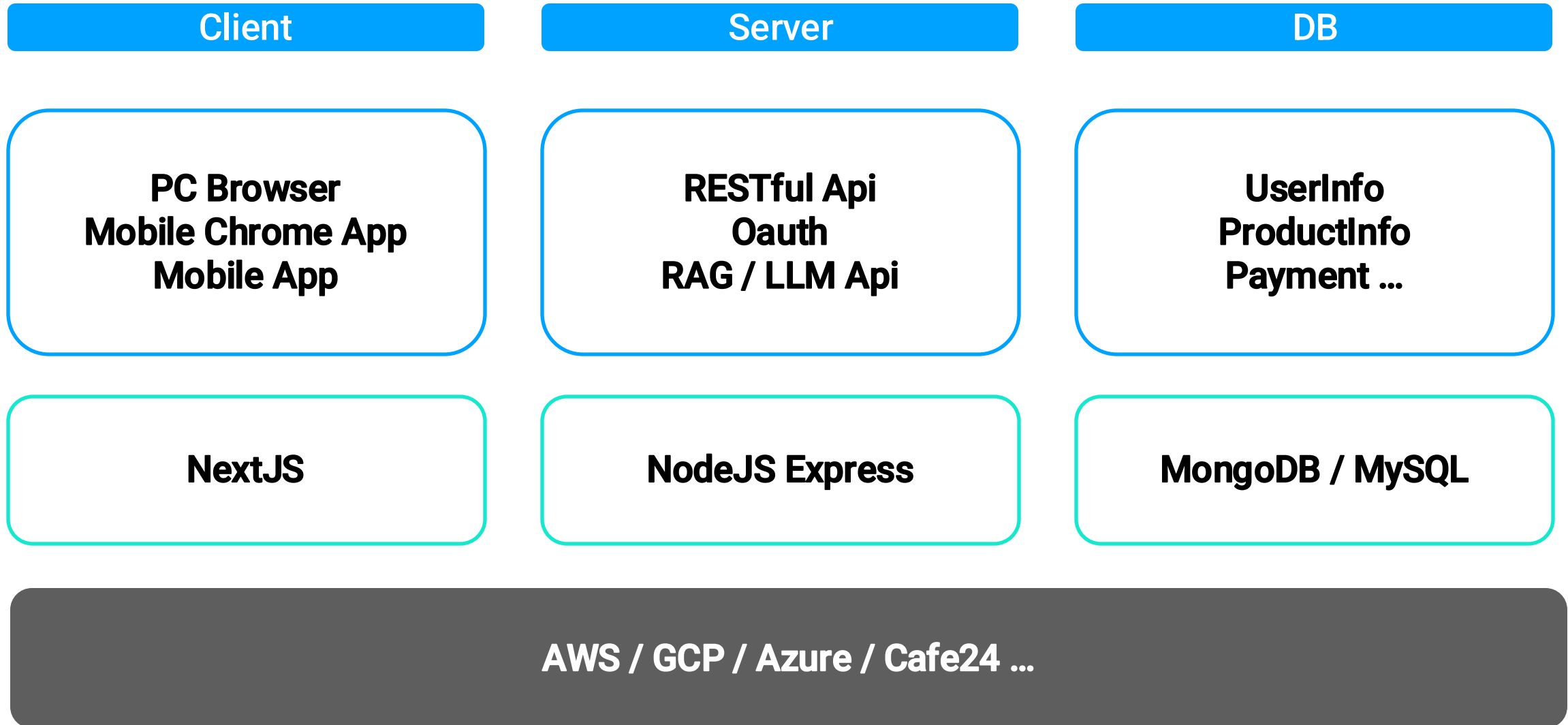
[Javascript, Python, Ruby, PHP, shell script]



Javascript로 시작하는 프로그래밍

- 웹 개발의 핵심 언어, 브라우저에서 바로 실행
- 프론트엔드(React, Vue)와 백엔드(Node.js) 모두 가능
- 초보자 친화적: 간단한 코드로 즉각적인 결과 확인





변수 선언, 할당, 초기화

변수를 생성하고 값을 저장하는 문법에서 var 키워드나 이후에 배우는 let, const 키워드를 사용 '변수의 식별자를 지정하는 행위를 **변수를 선언한다**고 부름.

할당 연산자인 = 기호로 우변에 있는 값을 변수 공간에 대입(저장)하는 것을 **값을 할당한다**고 함.

변수는 초기에 값을 할당하지 않고 선언만 할 수 있음.

var num;

식별자
↓
var num = 10 + 20;
↑ ↑ ↑
키워드 연산자 세미콜론

표현식 ← 값(30)

자료형

자료형(data type)이란 자바스크립트에서 사용할 수 있는 데이터의 종류를 의미.

자바스크립트의 자료형은 **기본 자료형**과 **참조 자료형**으로 구분.

기본(primitive) 자료형으로는 문자(string), 숫자(number), 논리(boolean), undefined, null, Symbol
자료형이 있고, 참조(reference) 자료형에는 객체(object)가 있음.

```
console.log(typeof 변수명);
```

감사합니다