02주. 데이터 준비와 연산 명령어로 기본 프로그램 만들기

02 자료형









학습내용

- 01 자료형(데이터 타입)의 필요성
- 02 자료형의 종류
- 03 자료의 형변환



학습목표

- 자료형(Data Type)의 개념과 필요성을 이해하고, 자료형이 프로그래밍에서 어떤 역할을 하는지 설명할 수 있다.
- JavaScript의 다양한 자료형(기본형과 참조형)을 구분하고, 각 자료형에 맞게 코드를 작성할 수 있다.
- JavaScript의 동적 타입 시스템(Dynamic Typing)의 특징을 이해하고, 변수의 자료형이 실행 중 적절하게 변경되도록 코드를 작성할 수 있다.

01

자료형(데이터 타입)의

필요성







프로그래밍 언어가 값을 어떻게 저장하고 처리할지를 결정할 수 있게 도와주어, 프로그램을 안정적으로 수행할 수 있도록 함

- 01 오류 방지
- 02 메모리효율성
- 03 명확한 연산 가능
- 04 디버깅용이

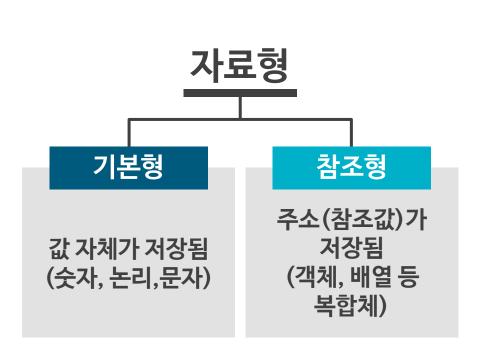
02

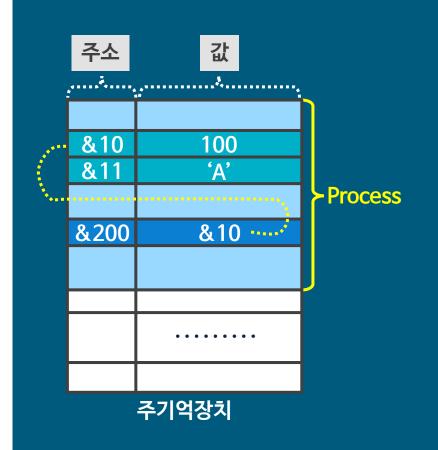
자료형의 종류



1) JavaScript의 자료형







JavaScript의 기본형(Primitive Type)



자료형	설명	예시
Number	숫자(정수, 실수)	42, 3.14
String	문자(열) 데이터	"hello", 'world'
Boolean	참/거짓	true, false
null	값이 없 음 을 명시	null
undefined	값이 할당되지 않음	undefined
BigInt	큰 정수(2 ⁵³ 이상)	12345678901234567890n
Symbol	고유하고 변경 불가능한 값	Symbol ('id')

※ 자료형 명칭은 래퍼클래스의 명칭으로 표기함

JavaScript의 참조형(Reference Type)



JavaScript에서 참조형은 본질적으로 모두 객체(Object)임

자료형	설명	예시	typeof 결과
Object	키-값 쌍을 저장하는 기본 객체형	{ name: "Lee", age: 25 }	"object"
Array	순서가 있는 값의 목록 (인덱스 기반)	[1, 2, 3]	"object"
Function	실행 가 능 한 코드 블록 (함수)	function greet() {}	"function"
Date	날짜와 시간 정보 객체	new Date()	"object"
RegExp	정규 표현식을 나타냄	/abc/i	"object"
Мар	키-값 구조로, 키에 어떤 값이든 가능	new Map([[1, 'one']])	"object"
Set	중복 없는 값을 저장하는 집합	new Set([1, 2, 2, 3])	"object"
WeakMap	키가 객체인 Map, 가비지 컬렉션 가능	new WeakMap()	"object"
WeakSet	객체만 저장 가능한 Set	new WeakSet()	"object"

03

자료의 형변환





"10" + 5

어떻게 자료형을 처리할까?



2) 동적 자료형 변환(Type Coercion)





변수 선언 시 자료형을 명시하지 않아도 되고, 실행 중 자료형이 변경될 수 있음



유연한 코딩, 빠른 개발이 가능함

유형	설명	예시
암시적 변환 (암 묵 적, 자동)	자바스크립트가 상황에 맞게 자동으로 자료형변환	"5"+ 1 ➡ "51"(문자열로 변환됨) "10" - 3 ➡ 7(숫자로 변환됨)
명시적 변환 (개발자 직접)	개발자가 함수로 자료형변환	Number("10") → 10 String(123) → "123" Boolean(0) → false

동적 형변환의 원칙



암묵적 형변환은 toPrimitive ➡ toString / toNumber / toBoolean

단계	설명	목적
toPrimitive	객체 ➡ 원시값으로 변환	연산 수행 전 우선 수행
toString	원시값 ➡ 문자열	문자열 연결(+) 시
toNumber	원시값 ➡ 숫자	산 술 연산(-, *, /) 시
toBoolean	값 ➡ true/false	조건문이나 논리연산 시

- [1] + 1 // → [1].toString() → "1" → "1" + 1 → "11"
- 01 [1] 인 배열객체를 문자열로 자동 변환 시도
- 02 [1].toString() 호출 ➡ "1"로 변환
- **03** "1" + 1→ "1" + "1"



● 연산자별 암묵적 형변환 규칙

+ 연산자

- 하나라도 문자열이면★ toString
- 둘 다 숫자/숫자형이면
 ➡ toNumber

<u>산술 연산자 -, *, /</u>

• 항상 toNumber

비교 연산자 〈, 〉, ==

- 양쪽 값을 toPrimitive 후 비교
- ==은 동등 비교 추상
 연산(Abstract Equality
 Comparison) 사용

```
let a = "5";
let b = 2;
console.log(a + b); // "52" → 문자열로 자동 변환(암시적)
console.log(Number(a) + b); // 7 ➡ 명시적으로 숫자로 변환
```

표현	변환 과정	결과
"5" + 1	"5" + "1"	"51"
"5" – 1	5 – 1	4
true + false	1 + 0	1
null + 1	0 + 1	1
[] + 1	"" + 1	"1"
{} + 1	"[object Object]" + 1	"[object Object] 1"
[] == 0	"" == 0 → 0 == 0	true



02주. 데이터 준비와 연산 명령어로 기본 프로그램 만들기

03 다양한 연산자



