02

값과 리터럴

학습내용

01 값(Value)

02 리터럴(Literal)

03 표현식(Expression)

04 문(Statement)

05 표현식인 문과 표현식이 아닌 문

학습목표

- JavaScript에서 값(Value)의 개념에 대해 설명할 수 있다.
- JavaScript에서 리터럴(Literal)의 개념을 이해하고, 다양한 리터럴의 종류를 예시를 통해 설명할 수 있다.
- 표현식(Expression)과 문(Statement)의 차이를 이해하고, 코드에서 각각을 구분할 수 있다.
- 표현식인 문(Expression Statement)과 표현식이 아닌 문(Declaration, Control Statement 등)의 차이를 설명하고, 실제 예제를 통해 구분할 수 있다.

01

값(Value)

1) 값(Value)이란?

값(Value)

값은 식(표현식, Expression)이 평가(Evaluate)되어 생성된 결과

참고 평가(Evaluate)

식을 해석해서 값을 생성하거나 창조하는 것

```
// 표현식
10 + 20 ;
```

평가(Evaluate) →

```
// 값 생성
30
```

2) 값과 변수의 관계

변수는 **하나의 값을 저장**하기 위해 확보한 메모리 공간 자체 또는 그 메모리 공간을 식별하기 위해 붙인 이름이라고 할 수 있음

```
// 변수 sum에는 10 + 20이라는 식이 평가되어 숫자 값 30이 할당됨
var sum = 10 + 20;    // var sum = 30;
```

02

리터럴 (Literal)

1) 리터럴(Literal)이란?

리터럴(Literal)

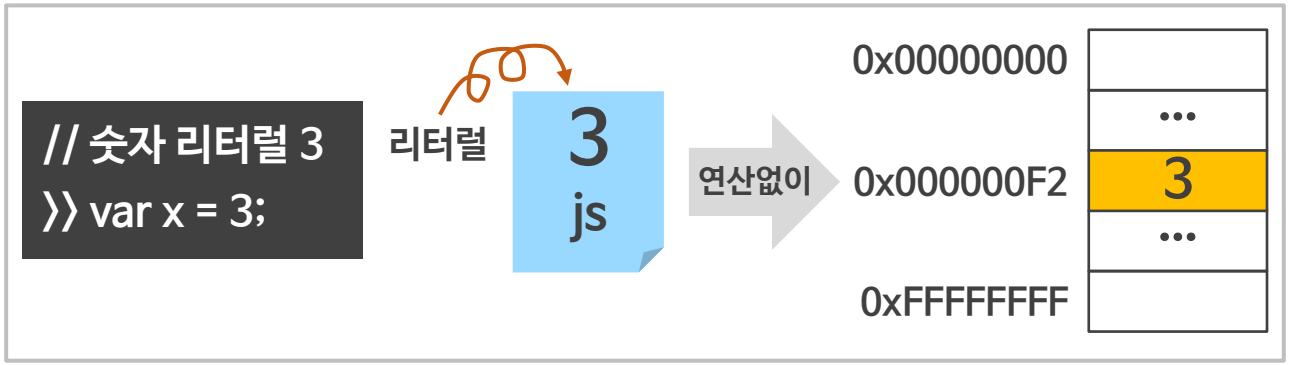
- 사람이 이해할 수 있는 문자(아라비아숫자, 알파벳, 한글 등) 또는 약속된 기호({},//,[] 등)를 사용해 값을 생성하는 표기법(Notation)

개발자가 코드에 직접 작성한 고정된 형태의 값 표현이며,
JavaScript 엔진에 의해 추가 연산 없이 즉시 사용될 수도 있고,
런타임 시 객체나 함수로 평가되어 생성될 수도 있음

리터럴은 **값**을 **직접** 생성하기 위해
개발자가 코드에 직접 작성한 고정된 형태의 값 표현

1) 리터럴(Literal)이란?

사람이 이해할 수 있는 아라비아 숫자를 사용해
숫자 리터럴 3을 코드에 기술하면 JavaScript 엔진은
추가 연산 없이 이를 **숫자 값 3을 생성**함

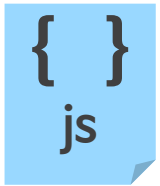


1) 리터럴(Literal)이란?

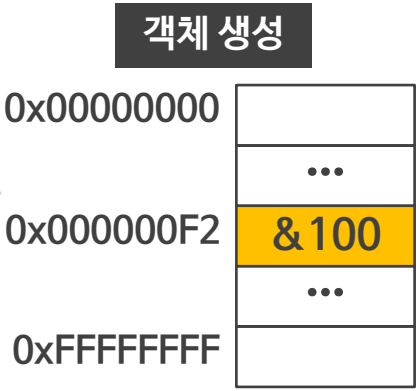
세종사이버대학교

객체 리터럴은 런타임에 새 객체를 생성하고, 키-값 쌍을
평가하여 등록한 후, 완성된 객체를 참조로 반환하는 표현식

```
const obj = {
  a: 1 + 2,
  b: Math.max(5, 10)
};
```



평가



〈평가순서〉

- {} → 새로운 빈 객체 생성
- a: 1 + 2 → 1 + 2 평가 → 3 → obj.a = 3 등록
- b: Math.max(5, 10) → 함수 호출 평가 → 10 → obj.b = 10 등록
- 전체 객체 { a: 3, b: 10 } → 변수에 할당

1) 리터럴(Literal)이란?

- 표현식과 리터럴

표현식(Expression)

- 값을 생성하는 문장
- 실행 시 평가(Evaluate)해서 값 생성
- 다른 표현식 안에 들어갈 수 있음

예 2 + 3, “hi” + “there”, Math.random()

리터럴(Literal)

- 개발자가 약속된 기호를 이용해 직접 값을 생성해 전달
- 코드 안에서 정해진 문법에 따라 **값 자체를 명시**한 것

예 5, “hello”, {a:1}, [1,2,3]

리터럴은 사실상 “값을 만드는 표현식의 단축 문법”

리터럴(literal)은 “값 그 자체를 코드 상에 명시적으로 작성한 것”

리터럴의 종류

리터럴 종류	설명	예시
숫자 리터럴 (Number)	정수 또는 실수를 표현	42, 3.14, -7, 0b1010, 0xFF
문자열 리터럴 (String)	큰따옴표(“), 작은따옴표(‘), 백틱(`)로 감싼 문자	“hello”, ‘world’, `hi`
불리언 리터럴 (Boolean)	참 또는 거짓을 표현	true, false
null 리터럴	의도적으로 “값 없음”을 표현	Null
undefined 리터럴	값이 정의되지 않음을 표현	Undefined
배열 리터럴 (Array)	대괄호 []로 감싸진 값의 리스트	[1, 2, 3], [], [true, “a”]

리터럴의 종류

리터럴 종류	설명	예시
객체 리터럴 (Object)	중괄호 {}로 감싼 키-값 쌍 구조	{name: “Lee”, age: 30}
함수 리터럴	직접 함수 정의	var f = function() { return 1; }
정규표현식 리터럴 (RegExp)	슬래시 /로 감싼 패턴	/abc/, /₩d+/g
템플릿 리터럴 (Template Literal)	백틱(``)과 ${} 표현 포함 문자열	`Hello, ${name}`
BigInt 리터럴	n 접미사를 붙여 큰 정수 표현	1234567890123456789 0n
Symbol 리터럴	고유한 식별자 생성	Symbol(‘id’)

3) 리터럴(Literal)의 특징

● 리터럴의 핵심 기능

> 리터럴은 값 그 자체를 표현하는 간결한 문법 구조로,
> 코드를 **읽기 쉽고** 작성 **효율을 높이는** 핵심요소

장점	설명	예시코드
간결성	값을 직접 표현하므로 짧고 명확한 코드 작성 가능	let num = 10; let name = “Alice”;
가독성	코드의 의도를 쉽게 파악할 수 있어 유지보수에 유리	let user = { id: 1, name: “Tom” };

성능
(정적 처리)
	컴파일러/인터프리터가 바로 인식 가능한 정적 값이므로 계산 비용 없음	let result = 100; //값
		VS
		let result = 10 * 10;

3) 리터럴(Literal)의 특징

● 객체 리터럴과 함수 리터럴의 기능

객체 리터럴	`let obj = { name: "Tom" };` • key-value 쌍으로 데이터를 직관적, 구조화하여 표현함(JSON과 호환) • 중첩된 복잡한 구조도 간결하게 표현함 • 메서드 축약 구문 등 현대적 문법 지원함 • 실제로는 런타임에서 새로운 객체가 생성됨
함수 리터럴	`let f = function(x) { return x * 2; };` • 변수에 할당, 인자로 전달, 반환 등 자유로운 조합 가능함 • 간결한 함수 정의 및 콜백, 클로저 활용 가능함 • JavaScript 엔진은 즉시 실행되지 않고, 평가 시점을 런타임으로 미룸 • 내부적으로는 Function 객체가 동적으로 생성됨 ➡ 동적 객체 기반 언어

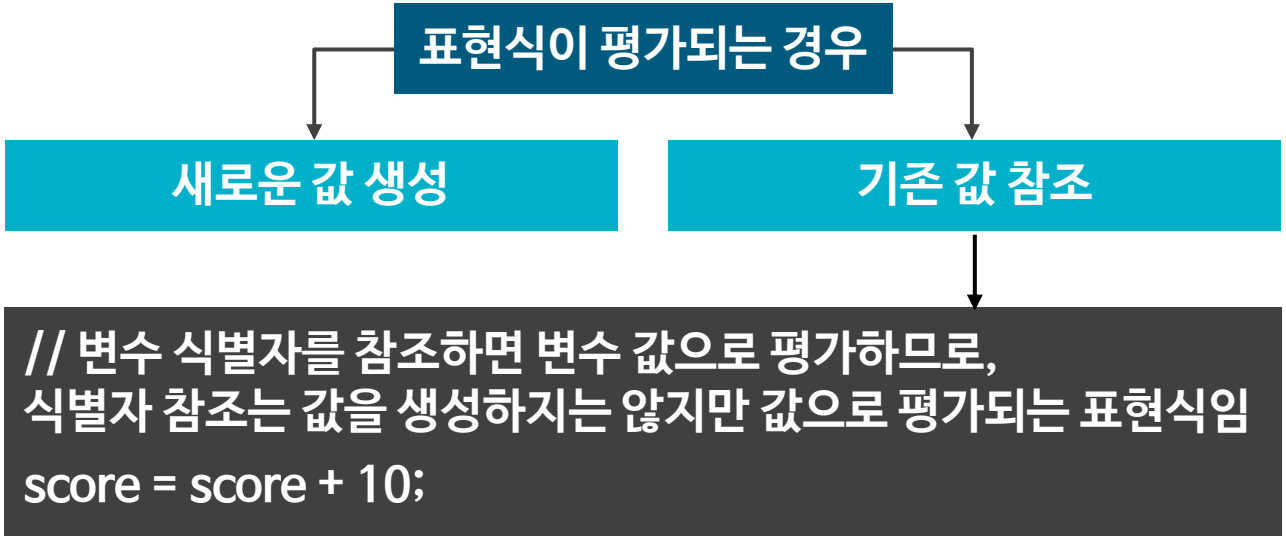
03

표현식 (Expression)

1) 표현식(Expression)이란?

표현식(Expression)

값으로 평가될 수 있는 문(Statement)



리터럴
Literal

값을 고정적으로 직접 입력한 것

예 숫자 5나 문자열 'hello'는 리터럴

그 자체로 값으로 평가됨

표현식
Expression

하나의 값으로 평가되는 코드 조각

예 3 + 5나 x * 2는 표현식

리터럴, 식별자(변수, 함수 등의 이름), 연산자, 함수 호출 등의 조합으로 구성

04

문(Statement)

1) 문(Statement)이란?

> 프로그램을 구성하는 기본 단위이자 최소 실행 단위

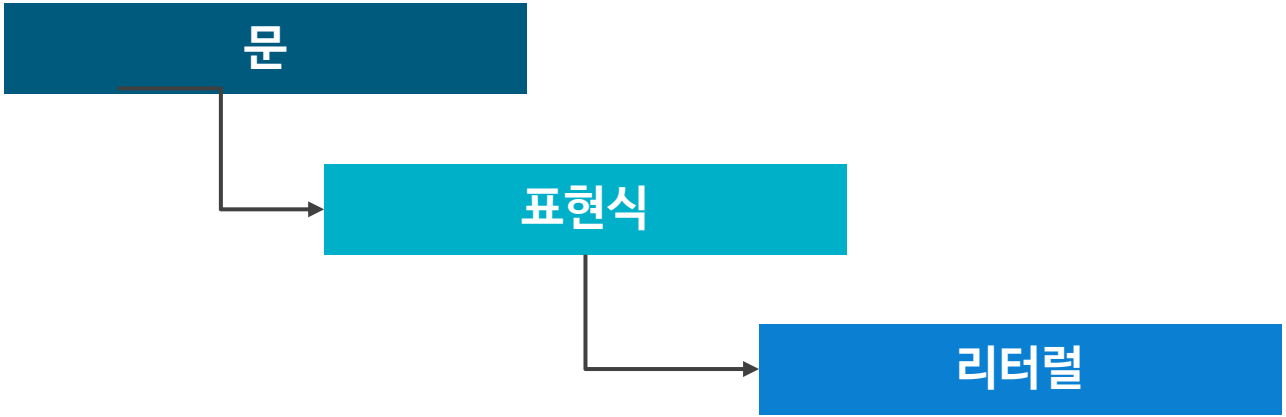
프로그램 = **문들의 집합**

예 명령문으로, 선언문, 할당문, 조건문, 반복문 등으로 구분됨

```
// 선언문
var  x;

// 할당문
x=5;

//함수 선언문
function foo() {   }
```

```
//조건문
if(x>1) { console.log("x"); }

//반복문
for(var  v in [10,20,30])
{
   console.log(v);
}
```

2) 문 & 표현식 & 리터럴

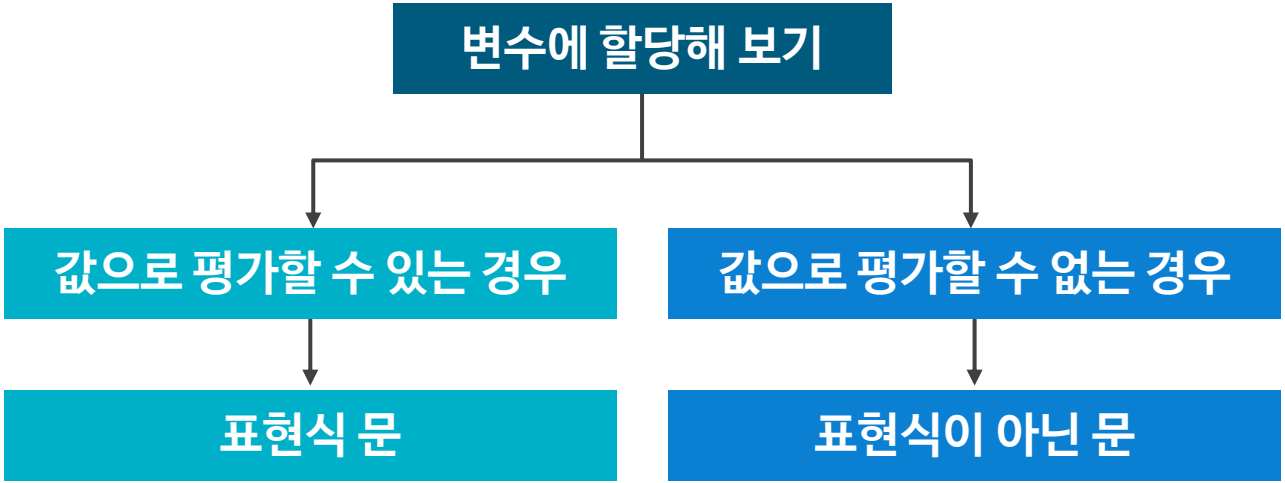


문	독립적인 명령 단위이고, 반드시 값으로 평가될 필요는 없음
표현식	값으로 평가되는 코드 조각임
리터럴	코드에 직접 쓴 고정된 값으로, 모든 리터럴은 표현식의 한 종류임

05

표현식인 문과
표현식이 아닌 문

1) 표현식 문과 표현식이 아닌 문의 구분 방법



예 var foo = var x; // var x는 선언문으로, 평가 불가 ➡ 표현식 아님

표현식이 아닌 문은 값처럼 사용할 수 없음

“ 표현식은 값으로 평가되는 코드 ”

```
// 변수 선언문은 표현식이 아닌 문이다.
var x;
```

표현식이 아닌 문은 디버깅도 할 수 없음

```
//할당문은 그 자체가 표현식이지만, 완전한 문이기도 하다. 즉, 할당문은 표현식인 문이다.
x= 100;
```

세종사이버대학교

04주. 효율적인 프로그래밍의 시작 : 함수

03

함수의 다양한 정의 방법