연산자(Operator)

연산자

- 연산의 사전적 의미
  - 규칙에 따라 계산하여 값을 구함
- 연산자Operator 형태
  - +, -, *, /, %
  - >, >=, <, <=
  - ==, !=, ===, !==
  - 콤마(,), typeof, delete, void
  - instanceof, in, new 등

표현식

- 표현식Expression 형태
  - 1 + 2
  - var total = 1 + 2;
  - var value = total / (2 + 3);
- "표현식을 평가"한다고 합니다.
- 표현식을 평가하면 결과가 반환되며
  - 이를 평가 결과라고 합니다.

할당 연산자

할당 연산자

- 단일 할당 연산자
  - = 하나만 사용
  - var result = 1 + 2;
- 복합 할당 연산자
  - = 앞에 연산자 작성
  - +=, -=, *=, /=, %=
  - <<=, >>=
  - >>>=, &=, ^=, |=
- 먼저 = 앞을 연산한 후, 할당
  - var point = 7;
  - point += 3;

해석, 실행 순서

- 해석이란?
  - JS 코드를 기계어로 바꾸는 것. Compile
  - "엔진Engine이 해석하고 실행한다"라고 합니다.
- 실행 순서
  - var result = 1 + 2 + 6;
  - = 왼쪽의 표현식 평가
  - = 오른쪽의 표현식 평가
    왼쪽에서 오른쪽으로 평가(1 + 2, 3 + 6)
  - = 오른쪽 표현식의 평가 결과를
    왼쪽 표현식 평가 결과에 할당

산술 연산자(+)

+ 연산자

- + 양쪽의 표현식을 평가
  - 평가 결과를 더함 코드 1
- 평가 결과 연결
  - 한 쪽이라도 숫자가 아니면 연결 코드 2
- 왼쪽에서 오른쪽으로 연산
  - 1 + 5 + "ABC" 결과는? 코드 3

숫자로 변환

숫자로 변환

- 연산하기 전에 우선 숫자로 변환
- 변환된 값으로 연산 코드 1 코드 2 코드 3

값 타입	변환 값
Undefined	NaN
Null	+0
Boolean	true: 1, false: 0
Number	변환 전/후 같음
String	값이 숫자이면 숫자로 연산 단, 더하기(+)는 연결

산술 연산자(-, *, /, %)

- 연산자

- 왼쪽 표현식 평가 결과에서
  오른쪽 표현식 평가 결과를 뺍니다.
- String 타입이지만,
  값이 숫자이면
  Number 타입으로 변환하여 계산 코드 1

* 연산자

- 왼쪽 표현식 평가 결과와
  오른쪽 표현식 평가 결과를 곱합니다.
- 숫자 값으로 변환할 수 있으면
  변환하여 곱합니다.
- NaN 반환
  - 양쪽의 평가 결과가
    하나라도 숫자가 아닐 때 코드 1
- 소수 값이 생기는 경우 처리 코드 2

/ 연산자

- 왼쪽 표현식 평가 결과를
  오른쪽 표현식 평가 결과로 나눕니다
- NaN 반환
  - 양쪽의 평가 결과가
    하나라도 숫자가 아닐 때
  - 분모, 분자 모두 0일 때
- 분모, 분자가 0일 때
  - 분모가 0이면 Infinity 반환
  - 분자가 0이면 0 반환
- [번개 코딩]
  - 위의 설명과 같은 코드를 작성하세요.

% 연산자

- 왼쪽 표현식 평가 결과를
  오른쪽 표현식 평가 결과로 나누어
  나머지를 구합니다.
- 3 % 2의 나머지는 1 코드 1

단항 연산자

단항 + 연산자

- 형태: +value
- 값을 Number 타입으로 변환 코드 1
- 코드 가독성
  - +를 더하기로 착각할 수도 있음
  - Number()도 기능 같음

단항 - 연산자

- 형태: -value
- 값의 부호를 바꿈
  - +는 -로, -는 +로 바꿈
- 연산할 때만 바꿈
  - 원래 값은 바뀌지 않음 코드 1

후치, 전치, 논리 **NOT** 연산자

후치 ++연산자

- 형태: value++
- 값을 자동으로 1 증가시킴
  - 문장을 수행한 후에 1 증가
  - 즉, 세미콜론(;) 다음에서 증가 코드 1

전치 ++연산자

- 형태: ++value
- 값을 자동으로 1 증가시킴
  - 문장 안에서 1 증가
  - 표현식을 평가하기 전에 1 증가
    표현식에서 증가된 값을 사용 코드 1

후치 --연산자

- 형태: value--
- 값을 자동으로 1 감소시킴
  - 문장을 수행한 후에 1 감소
  - 즉, 세미콜론(;) 다음에 감소 코드 1

전치 --연산자

- 형태: --value
- 값을 자동으로 1 감소시킴
  - 문장 안에서 1 감소
  - 표현식을 평가하기 전에 1 감소
    표현식에서 감소된 값을 사용 코드 1

! 연산자

- 논리Logical NOT 연산자
  - 형태: !value
- 표현식 평가 결과를 true, false로 변환한 후 true이면 false를, false이면 true를 반환
- 원래 값은 바뀌지 않으며 사용할 때만 변환 코드 1

유니코드, **UTF**

유니코드

- Unicode
  - 세계의 모든 문자를 통합하여 코드화
  - 언어, 이모지 😃 ☕ 등
  - 코드 값을 코드 포인트code Point라고 부름
  - 0000~FFFF, 10000~1FFFF 값에 문자 매핑
  - 유니코드 컨소시엄 http://www.unicode.org/
- 표기 방법
  - u와 숫자 형태: u0031은 숫자 1
  - JS는 u앞에 역슬래시(\) 작성 코드 1
  - 역슬래시(\)를 문자로 표시하려면 역슬래시(\\) 2개 작성 코드 2
  - ES6에서 표기 방법 추가됨

UTF

- Unicode Transformation Format
  - 유니코드의 코드 포인트를 매핑하는 방법
  - UTF-8, UTF-16, UTF-32로 표기
  - <meta charset="utf-8">
  - UTF-8은 8비트로 코드 포인트 매핑
  - 8비트 인코딩Encoding이라고 부름

관계 연산자

관계 연산자

- 관계Relational 연산자
  - <, >, <=, >= 연산자
  - instanceof 연산자
  - in 연산자
- instanceof, in 연산자
  - 사전 설명이 필요하므로
    관련된 곳에서 다룹니다.

> 연산자

- 부등호: Greater-than
- 양쪽이 Number 타입일 때
  - 왼쪽이 오른쪽보다 크면 true 반환
    아니면 false 반환 코드 1
- String 타입 비교
  - 한 쪽이 String 타입이면 false 코드 2
  - 양쪽이 모두 String 타입이면
    유니코드 사전 순서로 비교 코드 3
  - 문자 하나씩 비교 코드 4
- <, <=, >=는 비교 기준만 다름

동등, 부등, 일치, 불일치 연산자

== 연산자

- 동등 연산자
- 왼쪽과 오른쪽 값이 같으면 true
  다르면 false
- 값 타입은 비교하지 않음
  1과 "1"이 같음 코드 1 코드 2 코드 3

!= 연산자

- 부등 연산자
- 왼쪽과 오른쪽 값이 다르면 true
  같으면 false
- a != b와 !(a == b)가 같음

=== 연산자

- 일치 연산자
- 왼쪽과 오른쪽의 값과 타입이
  모두 같으면 true
  값 또는 타입이 다르면 false
- 1 === 1, true
  1 === "1", false 코드 1 코드 2 코드 3

!== 연산자

- 불일치 연산자
- 값 또는 타입이 다르면 true
  true가 아니면 false

콤마, 그룹핑, 논리 연산자

콤마 연산자

- 기호: , (Comma)
- 콤마로 표현식을 분리
  - var a = 1, b = 2;
  - 한 번만 var 작성

() 연산자

- 그룹핑 연산자
- 소괄호() 안의 표현식을 먼저 평가
  - 평가한 값을 반환
  - 5 / (2 + 3)

|| 연산자

- 논리 OR 연산자
- 표현식의 평가 결과가
  하나라도 true이면 true
  아니면 false 코드 1 코드 2
- 왼쪽 결과가 true이면
  오른쪽은 비교하지 않음 코드 3

&& 연산자

- 논리 AND 연산자
- 표현식의 평가 결과가
  모두 true이면 true
  아니면 false 코드 1
- 왼쪽 결과가 false이면
  오른쪽은 비교하지 않음 코드 2

조건 연산자, 연산자 우선순위

조건 연산자

- 기호: exp ? exp-1 : exp-2
  - 3항 연산자라고도 함
- exp 위치의 표현식을 먼저 평가
  - true이면 exp-1의 결과 반환
  - false이면 exp-2의 결과 반환 코드 1

연산자 우선순위

- 연산자의 실행 우선순위
  - ECMA-262 스펙에 없음
- 우선순위가 가장 높은 것은 ()
- MDN Operator precedence