연산자

1. 산술 연산자

연산자	설명
+	더하기
-	빼기
*	곱하기
/	나누기
~/	몫 구하기
%	나머지 값
++	현재 값에서 1 더하기
—	현재 값에서 1 빼기

2. 할당 연산자
    a. 기본 할당 연산자

연산자	사용	설명
=	변수 = 값	변수에 값을 할당
??=	변수 ??= 값	변수가 null일 경우에만 값을 할당

    b. 복합 할당 연산자

연산자	설명
+=	변수에 현재 값를 더한 후 변수에 다시 할당
-=	변수에 현재 값를 뺀 후 변수에 다시 할당
*=	변수에 현재 값를 곱한 후 변수에 다시 할당
/=	변수에 현재 값를 나눈 후 변수에 다시 할당
~/=	변수에 현재 값을 나눈 몫을 구한 후 변수에 다시 할당

3. 관계 연산자

연산자	설명
==	피연산자 A와 B가 같다
!=	피연산자 A와 B가 같지 않다
>	피연산자 A가 B보다 크다
<	피연산자 A가 B보다 작다
>=	피연산자 A가 B보다 크거나 같다
<=	피연산자 A가 B보다 작거나 같다

4. 논리연산자

연산자	의미	설명
!	NOT	해당 값을 부정 (true ⇒ false / false ⇒ true)
&&	AND	피연산자 모두 true일 경우만 true
||	OR	피연산자 모두 false일 경우만 false

5. 비트 & 시프트 연산자
    - 참고사이트 : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators
    - 숫자를 이루는 각각의 비트를 조작하는 연산자

    > 진수 표현법

```
var valA = 20;      // 10진수
var varB = 0x014;   // 16진수
```

연산자	의미	설명
&	AND	비트 단위로 둘 다 1일 경우 1
|	OR	비트 단위로 둘 다 0일 경우 0
^	XOR	비트 단위로 값이 다를 경우 1
<<	←	시프트 값을 왼쪽으로 옮김
>>	→	시프트 값을 오른쪽으로 옮김

```
int a = 5; // 0000101
int b = 3; // 0000011

print(a & b);  // 0000001
print(a | b);  // 0000111
print(a ^ b);  // 0000110
print(a >> b); // 0000000
print(a << b); // 0101000
```

```
1
7
6
0
40
```

6. 타입검사연산자

연산자	설명
as	타입변환, 다른 타입으로 변환은 되지 않고 상위 타입으로 변환할 수 있음
is	특정 객체가 특정 타입이면 true
is!	특정 객체가 특정 타입이 아니면 true

```
class Person {
  var name = 'Person';
}

class Employee extends Person {
  var name = 'Employee';
}

class Student extends Person {
  var name = 'Student';
}

main() {
  Employee emp = new Employee();
  Student std = new Student();

    //타입변환
  Person first = emp as Person;
```

```
Person second = std as Person;
// Person person = emp; // as Person은 생략 가능

print('first.name = ${first.name}');
print('second.name = ${second.name}');

print('(emp as Person).name = ${(emp as Person).name}');
// print('(emp as Student).name = ${(emp as Student).name}'); // typeError;

  // 타입 검사 : is
if (emp is Employee) {
  print('emp is Employee');
} else {
  print('emp is not Employee');
}

  // 타입 검사 : is!
if (emp is! Employee) {
    print('emp is not Employee');
  } else {
    print('emp is Employee');
  }
}
```

7. 조건표현식
   a. 삼항연산자
      - 조건 ? 표현식1 : 표현식2
        ⇒ 만약 조건이 true 이면 표현식1의 값을 반환하고 false이면 표현식2의 값을 반환

```
var a = 100;
// 삼항 연산자
var result = (a >= 0) ? '양수' : '음수';
print(result);

// if문
if (a >= 0) {
  result = '양수';
} else {
  result = '음수';
}
print(result);
```

   b. 조건적 멤버 접근
      - 객체 ?. 멤버
        ⇒ 만약 객체가 null이면 null을 반환하고 null이 아니면 멤버 값을 반환

```
main() {
var emp = null;
// 조건적 멤버 접근
var result = emp?.name;
print(result);

// if문
if (emp != null) {
result = emp.name;
```

```
} else {
result = null;
}
print(result);
}

class Employee {
var name = 'Employee';
}
```

c. 널 확인 연산자

- 좌항 ?? 우항
  ⇒ 만약 좌항이 null이 아니면 좌항을 반환하고 null이면 우항의 값을 반환

```
main() {
  var emp = new Employee();
  // 널 확인 연산자
  var result = emp.name ?? 'No employee';
  print(result);

  // if문
  if (emp.name != null) {
    result = emp.name;
  } else {
    result = 'No employee';
  }
  print(result);
}

class Employee {
  var name = 'Employee';
}
```

8. 캐스케이드 표현법

a. 한 객체로 해당 객체의 속성이나 멤버 함수를 연속으로 호출할 때 유용

```
class Employee {
  var name = "employee";
  int? age;

  setAge(int age) {
    this.age = age;
  }

  showInfo() {
    print("$name is $age.");
  }
}

main() {
  // 캐스케이드 표기법 미사용
  Employee hong = Employee();
  hong.name = "Hong";
  hong.setAge(25);
  hong.showInfo();
```

```
  // 캐스케이드 표기법 사용
  Employee kang = Employee()
    ..name = 'Kang'
    ..setAge(25)
    ..showInfo();
}
```