변수와 상수

데이터를 저장할 수 있는 메모리 상의 공간

#01. 전역 변수

프로그램의 어느 곳 에서나 식별할 수 있는 변수를 의미

1) 변수의 선언과 할당

변수의 선언

- 컴퓨터 메모리(RAM)상에 데이터를 기록하기 위해 공간을 예약하고 그 공간을 식별할 수 있는 이름을 지정하는 처리로 이해할 수 있다.
- 변수 이름 앞에 `var` 키워드를 명시하고 변수 이름을 지정
- 하나의 구문은 세미콜론으로 끝나야 함.

```
var 변수이름;
```

값의 할당

- 선언된 변수에 대입연산자 =를 사용하여 값을 대입하는 처리를 **할당**이라고 한다.
- 항상 **오른쪽에서 왼쪽으로 대입**

```
변수이름 = 값;
```

2) 선언과 할당의 통합

- 변수를 선언하면서 대입 연산자를 활용하여 값의 할당까지 한 라인에서 처리한다.

```
var 변수이름 = 값;
```

3) 변수값 변경하기

- 한 번 값이 할당된 변수는 다른 값으로 새롭게 할당 가능

```
var num = 100;
num = 200;
```

4) 변수이름 규칙

1. 영어, 숫자, 언더바(_), $ 기호만 사용할 수 있다.
2. 첫 글자는 숫자로 시작할 수 없다.

> 일반적으로 영어 소문자로 시작한다.

3. 두 개 이상의 단어를 결합하여 이름을 지정하는 경우
    1. **스네이크 표기법**: 띄어쓰기가 필요한 위치에서 언더바(_)를 사용.
        - 변수를 정의할 때는 잘 사용되지 않는다.

        > home + work ==> home_work

    2. **카멜 표기법**: 띄어쓰기가 필요한 위치의 첫 글자를 대문자로 변경.
        - **변수** 정의할 때를 포함하여 거의 대부분의 경우 일반적으로 사용한다.

        > home + work ==> homeWork

잘 알려지지 않은 규칙이지만 **UTF-8** 환경에서는 **한글도 사용할 수 있다.**

5) 변수의 재선언

- `var` 키워드를 사용하여 선언된 변수는 **중복 선언**이 가능
- 이후 뒤에서 공부할 **변수의 스코프(유효성 범위)** 도 무시된다.
- ***이러한 특성은 JS를 제외한 모든 프로그래밍 언어들의 규칙에는 위배되는 사항이므로 가급적 사용하지 않는 것이 좋다.***

> var 키워드를 사용을 자제하는 것이 좋다.

#02. 지역변수

- ES6 버전에서 새로 추가된 변수 생성 방법.
- 대부분의 프로그래밍 언어에서 말하는 일반적인 변수의 생성 규칙을 따른다.

> 적극 사용을 권장함.

1) 변수 선언, 할당

선언

- `let` 키워드를 사용하여 선언한다.

```
let 변수이름;
```

할당

- `var` 키워드를 사용한 전역 변수의 경우와 동일하다.

```
변수이름 = 값;
```

2) 선언과 할당의 통합

- `var` 키워드의 경우와 동일한 규칙으로 선언과 할당을 한 행에 축약할 수 있다.

```
let 변수이름 = 값;
```

3) 중복 선언 금지

- `let` 키워드를 사용하여 선언된 변수는 중복 선언 불가능.
- `let` 키워드를 통해 생성된 변수는 **일반적인 프로그래밍 언어의 규칙을 모두 따른다.**

#03. 상수

- 최초로 값을 할당한 이후 값을 변경할 수 없는 형태. (=읽기전용)
- `const` 키워드를 사용
- **선언과 동시에 값이 할당**되어야 함.
- 상수의 이름은 대문자만으로 구성된 스네이크 표기법을 사용하여 생성하는 것이 일반적.

#04. 변수의 자료형 (데이터 타입)

1) 변수에 저장할 수 있는 값의 종류

자바스크립트의 데이터 타입은 변수에 값을 할당(대입)할 때 결정된다.

타입	설명
number	정수와 실수를 포함하는 모든 숫자 형태
string	문자열. 쌍따옴표나 홑따옴표의 쌍으로 감싼 모든 형식의 데이터
boolean	논리형. true(참) 혹은 false(거짓)
object	객체. 함수(Function), 배열 (Array), 날짜 (Date), 정규식 (RegExp) 등을 포함한다
null	object 형의 한 종류. 나중에 할당하기 위해 임의로 비워 놓은 상태를 의미하는 특수한 값.
undefined	정의되지 않음. 선언만 하고 값이 할당되지 않은 상태

2) 변수의 자료형 확인하기

- `typeof` 연산자는 피연산자의 평가 전 자료형을 나타내는 문자열을 반환한다.

```
let a = 100;
console.log(typeof a);
```

#05. 형식문자(Format Character)

데이터를 입/출력 할 때, 컴퓨터가 그 Type을 인식할 수 있도록, 데이터의 해석을 지시해주는 문자

형식 문자가 사용되는 이유는, 같은 데이터라도 해석하는 방향에 따라 다른 문자가 될 수 있기 때문이다. 예로, 'a'라는 알파벳 소문자 데이터는 글자로 인식하는 경우(%c) a를 출력하지만, 정수로 인식하는 경우(%d) 97을 출력할 수 있다.

형식문자	설명
%d	모든 종류의 숫자
%s	글자, 문장
%o	객체. Javascript의 모든 데이터는 객체로서 존재하기 때문에 사실상 모든 종류의 값에 적용 가능
%j	JSON객체. JSON역시 객체의 한 종류이므로 사실상 모든 종류의 값에 적용 가능

```
console.log("%s %d", "Hello", 123);
```

#06. 변수 출력하기

자바스크립트 문자열을 역따옴표로 감싸고 그 안에서 변수나 표현식을 ${}로 감싸서 표현한다.

아래의 코드는 `Hello Javascript`가 출력된다.

```
const a = "Hello";
console.log(`${a} Javascript`);
```