## JavaScript 문법과 타입

### 목표

JavaScript의 기본 문법과 변수 선언, 데이터 형 및 리터럴을 다룹니다.

### 기본

- 명령을 문(statement)이라고 부르며, 세미콜론(;)으로 분리

- 세미콜론을 추가해 문을 끝내기를 권장

### 선언

- 기본적으로 3가지 방법이 있음

- var

+ 변수를 선언. 추가로 동시에 값을 초기화

- let

+ 블록 범위(scope) 지역 변수를 선언. 추가로 동시에 값을 초기화

- const

+ 읽기 전용 상수를 선언

### 변수

- 변수명은 식별자(identifier)라고 불리며 특정 규칙을 따름

- 식별자

+ 변수나 함수에 이름을 붙이거나 자바스크립트 코드 내 루프 문에 레이블을 붙이는 데 사용

+ 첫번째 문자는 알파벳(letter), 밑줄(\_) 혹은 달러 표시($)

+ 이어지는 문자들은 알바벳(letter), 숫자, 밑줄(\_) 혹은 달러 표시

+ 숫자는 첫 번째 문자로 허용되지 않음

+ 밑줄(\_) 혹은 달러 표시($)를 제외한 특수문자와 공백은 사용할 수 없음

+ I, my\_variable\_name, v13, \_dummy, $str

- 변수는 보통 소문자 낙타표시법을 사용해서 표현

#### 변수 선언

- var: var x = 42

+ 지역 및 전역 변수를 선언

- x = 42 로 선언 → 전역 변수로 지정

+ strict mode의 JavaScript에서 경고를 발생시켜 실행이 안됨

+ [strict mode]( https://msdn.microsoft.com/ko-kr/library/br230269(v=vs.94).aspx)

- let: let y = 13

+ 블록 범위 지역 변수를 선언하는데 사용

#### 변수 평가

- 초기값 없이 var 혹은 let 문을 사용해서 선언된 변수는 undefined

- 선언되지 않은 변수에 접근을 시도하는 경우 ReferenceError 예외

- undefined: boolean 문맥에서 사용될 때 false로 동작

- undefined: 수치 문맥에서 사용될 때 NaN으로 변환

- null: 수치 문맥에서는 0으로, boolean 문맥에서는 false로 동작

#### 변수 범위

- 전역변수: 어떤 함수의 바깥에 변수를 선언하면, 현재 문서의 다른 코드에 해당 변수를 사용

- 지역변수: 함수 내부에 변수를 선언하면, 오직 그 함수 내에서만 사용

#### 변수 호이스팅

- 예외를 받지 않고도, 나중에 선언된 변수를 참조할 수 있다는 것

#### 전역 변수

- global 객체의 속성(property)입니다. 웹 페이지에서 global 객체는 window 이므로, windows.variable 구문을 통해 전역 변수를 설정하고 접근가능

### 상수

- const 키워드로 읽기 전용 상수

- 변수 식별자와 같음

- 스크립트가 실행 중인 동안 대입을 통해 값을 바꾸거나 재선언될 수 없습니다.

- 범위 규칙은 let 블록 범위 변수와 동일

- 같은 범위에 있는 함수나 변수와 동일한 이름으로 선언할 수 없음

### 데이터 구조 및 형

#### 데이터 형

- ECMAScript 표준은 7가지 데이터 형을 정의

- 6가지 원시 데이터 형

+ Boolean. true와 false

+ null. null 값을 나타내는 특별한 키워드. JavaScript는 대소문자를 구분하므로, null은 Null, NULL 혹은 다른 변형과도 다릅니다.

+ undefined. undefined 값인 최상위 속성.

+ Number. 42 혹은 3.14159.

+ String. "안녕"

+ Symbol. (ECMAScript 6에 도입) 인스턴스가 고유하고 불변인 데이터 형

+ [Symbol설명](<https://msdn.microsoft.com/ko-kr/library/dn919632(v=vs.94).aspx)>

+ [Symbol설명]( http://hacks.mozilla.or.kr/2015/09/es6-in-depth-symbols/)

- Object

#### 데이터 형 변환

- 변수를 선언할 때 데이터 형을 지정할 필요가 없음

- 데이터 형이 스크립트 실행 도중 필요에 의해 자동으로 변환됨

- 숫자와 문자열 값 사이에 + 연산자를 포함한 식에서, JavaScript는 숫자 값을 문자열로 변환

- 다른 연산자를 포함한 식의 경우, JavaScript는 숫자 값을 문자열로 변환하지 않음

#### 문자열을 숫자로 변환하기

- parseInt(): 정수만 반환

+ [설명](<https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/parseInt>)

- parseFloat()

+ [설명](<https://www.w3schools.com/jsref/jsref_parsefloat.asp>)

### 리터럴

- JavaScript에서 값을 나타내기 위해 리터럴을 사용합니다. 이는 말 그대로 스크립트에 부여한 고정값임

#### 배열 리터럴

- 0개 이상의 식(expression) 목록

- 각 식은 배열 요소를 나타내고 대괄호([])로 묶임

- 배열 리터럴을 사용하여 배열을 만들 때, 그 요소로 지정된 값으로 초기화되고, 그 길이는 지정된 인수의 갯수로 설정

#### 불린 리터럴

- 불린 형은 두 리터럴 값을 가집니다. true와 false

- 원시 불린 값 true 및 false와 Boolean 객체의 true 및 false 값은 다름

- [설명](<http://stackoverflow.com/questions/856324/what-is-the-purpose-of-new-boolean-in-javascript>)

#### 정수

- 10진수: 선행 0(zero)이 아닌 숫자열

- 8진수: 선행 0(zero)이나 선행 0o(혹은 0O)

- 16진수: 선행 0x(나 0X)

- 2진수: 선행 0b(나 0B)

#### 객체 리터럴

- 중괄호({})로 묶인 0개 이상인 객체의 속성명과 관련 값 쌍 목록

- 문의 시작에 객체 리터럴을 사용해서는 안됩니다. 이는 {가 블록의 시작으로 해석되기 때문에 오류를 이끌거나 의도한 대로 동작하지 않음

#### 정규식 리터럴

- 슬래시 사이에 감싸인 패턴

#### 문자열 리터럴

- 큰 따옴표(") 혹은 작은 따옴표(')로 묶인 0개 이상의 문자

- 문자열 객체의 모든 메서드를 호출

- 문자 이스케이프

+ 백슬래시를 이용해 백슬래시 뒤에 있는 것을 문자 그대로 이해하겠다.