웹 앱 개발을 위한 Javascript 기초

2. 연산자와 명령문

한국기술교육대학교
온라인평생교육원

학습내용

1. 기본 명령어
2. 연산자

학습목표

1. 자바스크립트에서 사용하는 기본 용어에 대해 설명할 수 있다.
2. 기본 출력 방법에 대해 설명할 수 있다.
3. 기본 자료형, 변수, 상수를 이해하고, 관련된 연산자의 사용 방법에 대해 설명할 수 있다.
4. 자료형 변환 방법에 대해 설명할 수 있다.

1. 기본 명령어

표현식

a console
A Console
} 각각 다른 문자열로 처리

```
273;
let name = “홍”+“길”+“동”
console.log(“hello”)
```

문장

* 표현식이 하나 이상 모일 경우, 마지막에 종결 의미로 세미콜론(;)

프로그램

* 문장이 모이면 프로그램이 됨
* 문장을 작성하고, 다음 줄에 문장을 작성하면 앞 문장 끝에 세미콜론이 없어도 자바스크립트 엔진이 자동으로 세미콜론 추가

1. 기본 명령어

프로그램

* 자바스크립트 엔진이 세미콜론을 생략한 줄이 다음 줄과 이어지고 있다고 판단하면?

세미콜론을 자동으로 추가하지 않음

```
console.log("hello")
console.log("hi")
```

↓

```
console.log("hello");
console.log("hi");
```

```
c = a + b
(x+y).toString()
```

↓

```
c = a + b(x+y).toString()
```

예약어

* 자바스크립트 문법을 규정짓기 위해 자바스크립트 언어 사양에서 사용하는 특수한 키워드는 식별자로 사용하지 않는 편이 좋음

ECMAScript 6의 예약어

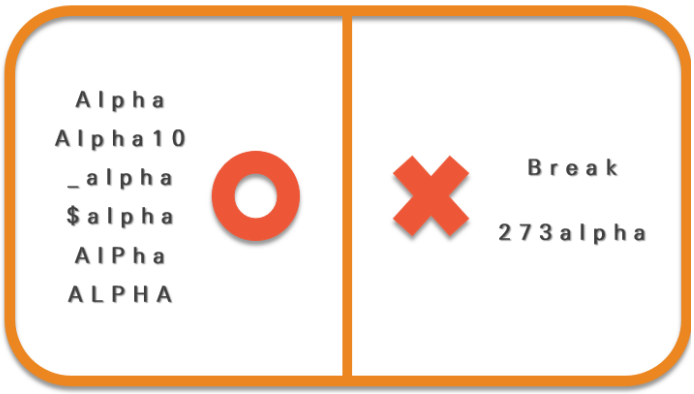
break	case	catch	class	const	continue
debugger	default	delete	do	else	export
extends	false	finally	for	function	if
import	in	instanceof	new	null	return
super	switch	this	throw	true	try
typeof	var	void	while	with	yield

* 현재는 약어가 아니지만 향후에 ECMAScript 확장을 위해 예약된 키워드
: await, enum, implements, package, protected, interface, private, public

1. 기본 명령어

식별자

- 이름을 붙일 때 사용하는 단어, 변수와 함수 이름 등으로 사용
- 특수 문자는 _와 $만 허용
- 숫자로 시작하면 안됨
- 공백은 입력하면 안됨



- 식별자 사용 규칙
  - 생성자 함수의 이름은 항상 대문자로 시작
  - 변수, 함수, 속성, 메소드의 이름은 항상 소문자로 시작
  - 여러 단어로 된 식별자는 각 단어의 첫 글자를 대문자로 함

Ex) will out ⇨ willOut
will return ⇨ WillReturn
i am a boy ⇨ iAmABoy

1. 기본 명령어

* 식별자의 종류

구분	단독으로 사용	다른 식별자와 사용
식별자 뒤에 괄호 없음	변수 또는 상수	속성
식별자 뒤에 괄호 있음	함수	메소드

```
alert('Hello World')      →  함수
Array.length              →  속성
input                     →  변수 또는 상수
propmt('Message','Defstr') →  함수
Math.PI                   →  속성
Math.abs(-273)            →  메소드
```

* 주석 프로그램의 진행에 영향을 주지 않는 코드

방법	표현
한 줄 주석 처리	// 주석
여러 줄 주석 처리	/* 주석 주석 */

```
// 주석은 코드의 실행에 영향을 주지 않습니다.
/*
console.log("JavaScript Programming")
console.log("JavaScript Programming")
console.log("JavaScript Programming")
*/
```

1. 기본 명령어

문자열 집합

* 문자열 생성 시 큰따옴표나 작은따옴표를 사용

```
console.log("This is 'String'")
This is 'String'
undefined
console.log('This is "String"')
This is "String"
undefined
```

문자열 집합

* 자바스크립트를 HTML 요소에 끼워 넣을 때는 자바스크립트 프로그램을 문자열로 작성

〈input type=“button” value=“Click” onclick=“alert(‘Thanks!’)”〉

※ HTML 코드에는 큰따옴표를 사용하지 않고 자바스크립트 코드에는 작은따옴표를 사용하여 HTML과 자바스크립트에서 사용하는 따옴표를 구분하는 것이 좋음

1. 기본 명령어

이스케이프 문자

* 따옴표를 문자 그대로 사용 가능
* 문자열 줄 바꿈 할 경우 사용

```
console.log("이름\t나이")
이름	나이
undefined
console.log("안녕\n하세요")
안녕
하세요
undefined
console.log("\\\\")
\\
undefined
```

```
console.log('This is\"String"')
This is"String"
undefined
console.log("This is\'String'")
This is'String'
undefined
```

* 이스케이프 시퀀스

이스케이프 시퀀스	의미
\0	널(null) 문자
\b	백스페이스 문자
\t	수평 탭 문자
\n	개행 문자
\v	수직 탭 문자
\f	다음 페이지 문자
\r	캐리지 리턴 문자(CR)
\'	작은따옴표 문자
\"	큰따옴표 문자
\\	역슬래시 문자

이스케이프 시퀀스	의미
\xXX	두 자릿수 16진수 XX로 지정된 Latin-1 문자
\uXXXX	네 자릿수 16진수 XXXX지정된 유니코드 문자
\u{XXXXXX}	16진수 코드포인트 XXXXXX로 지정된 유니코드 문자

1. 기본 명령어

문자열 합치기

연산자	설명
+	문자열 연결 연산자

```
console.log("가나다"+"라마"+"바사아")
가나다라마바사아
undefined
```

상수

- '항상 같은 수'라는 의미 ⟷ '변수'
- const – 상수(constant)를 만드는 키워드
- 변하지 않을 대상에 상수를 적용

변수

- 값을 저장할 때 사용하는 식별자, 변수 선언 후 변수에 값을 할당

변수 선언

```
let 식별자;
```

변수 pi를 선언

```
〉let pi;
undefined
```

변수 pi에 값을 할당

```
〉let pi;
undefined
〉pi = 3.14159265;
undefined
```

변수 초기화

```
〉let pi = 3.14159265;
undefined
```

2. 연산자

사칙연산

연산자	설명
+	덧셈 연산자
-	뺄셈 연산자
*	곱셈 연산자
/	나눗셈 연산자

```
25+50
75
50-20
30
10*3
30
50/10
5
```

```
console.log(25)
25
undefined
console.log(25.432)
25.432
undefined
```

* 연산자 우선순위

우선순위	연산자	결합 법칙
1	( ) (그룹연산자)	없음
2	. , [] new(인수 있음)	왼쪽 → 오른쪽 오른쪽 → 왼쪽
3	( ) (함수호출) new(인수없음)	왼쪽 → 오른쪽 오른쪽 → 왼쪽
4	++(후위), --(후위)	없음
5	!, ~, +(단항), -(부호 반전), typeof, void, delete, ++(전위), --(전위)	오른쪽 → 왼쪽

2. 연산자

연산자 우선순위

우선순위	연산자	결합 법칙
6	*, / , %	왼쪽 → 오른쪽
7	+, −, +(문자열 결합)	왼쪽 → 오른쪽
8	〈〈, 〉〉, 〉〉〉	왼쪽 → 오른쪽
9	〈, 〈=, 〉, 〉=, in, instanceof	왼쪽 → 오른쪽
10	==, !=, ===, !==	왼쪽 → 오른쪽

연산자 우선순위

우선순위	연산자	결합 법칙
11	&	왼쪽 → 오른쪽
12	^	왼쪽 → 오른쪽
13	|	왼쪽 → 오른쪽
14	&&	왼쪽 → 오른쪽
15	||	왼쪽 → 오른쪽

2. 연산자

* 연산자 우선순위

우선순위	연산자	결합 법칙
16	?:	오른쪽 → 왼쪽
17	yield,yield*	오른쪽 → 왼쪽
18	=, +=, −=, *=, /=, %=, ⟨⟨=, ⟩⟩=, ⟩⟩⟩=, &=, ^=, |=	오른쪽 → 왼쪽
19	...	없음
20	,	왼쪽 → 오른쪽

* 비교 연산자

연산자	설명
==	같습니다.
!=	다릅니다.
⟩	왼쪽 피연산자가 큽니다.
⟨	오른쪽 피연산자가 큽니다.
⟩=	왼쪽 피연산자가 크거나 같습니다.

```
52<273
true
52>273
false
```

2. 연산자

* 논리 연산자

연산자	설명
!	논리 부정 연산자
||	논리합 연산자
&&	논리곱 연산자

```
console.log(!true)
false
undefined
console.log(!(52<273))
false
undefined
```

문자열 합치기

* 변수에 사용할 수 있는 몇 개의 특별한 연산자가 존재
* 결과 값 : a+ = 10 는 a = a+10

복합 대입 연산자

숫자

연산자	설명
+=	숫자 덧셈 후 대입 연산자
-=	숫자 뺄셈 후 대입 연산자
*=	숫자 곱셈 후 대입 연산자
/=	숫자 나눗셈 후 대입 연산자

문자

연산자	설명
+=	문자열 연결 후 대입 연산자

2. 연산자

증감 연산자

- 변수 number를 초기화 하고 ++ 연산자와 -- 연산자를 사용
- 각 연산자에서 변수 값이 1만큼 변경됨
- 전위는 문장을 실행하기 전에 값을 변경하라는 의미
- 즉, console.log (++number) 코드는 console.log (number)를 실행하기 전에 변수 number에 1을 더함

연산자	설명
변수++	기존 변수 값에 1을 더합니다.(후위)
++변수	기존 변수 값에 1을 더합니다.(전위)
변수--	기존 변수 값에서 1을 뺍니다.(후위)
--변수	기존 변수 값에서 1을 뺍니다.(전위)

```
let number=10;number++;console.log(number);number--;console.log(number);
11
10
undefined
```

자료형 확인 연산자

연산자	설명
typeof	해당 변수의 자료형을 추출합니다.

```
typeof(10)
"number"
typeof("문자열")
"string"
```

Undefined 자료형

- 변수를 선언했으나 초기화 하지 않은 자료형

2. 연산자

강제 자료형 변환

함수	설명
Number()	숫자로 자료형 변환합니다.
String()	문자열로 자료형 변환합니다.
Boolean()	불로 자료형 변환합니다.

```
console.log(Number("52"));console.log(Number(true));console.log(Number("안녕하세요"));
52
1
NaN
undefined
```

* 숫자로 변환할 수 없는 문자열을 Number() 함수로 변환하면 NaN을 출력

* NaN(Not a Number)은 숫자 자료형이지만 숫자가 아닌 것을 의미

정리하기

1. 기본 명령어

- 표현식, 문장, 프로그램
- 예약어 : 자바스크립트 문법을 규정짓기 위해 자바스크립트 언어 사양에서 사용하는 특수한 키워드는 식별자로 사용하지 않는 편이 좋음
- 식별자 : 이름을 붙일 때 사용하는 단어, 변수와 함수 이름 등으로 사용
- 문자열 : 자바스크립트를 HTML 요소에 끼워 넣을 때는 자바스크립트 프로그램을 문자열로 작성
- 변수 : 값을 저장할 때 사용하는 식별자, 변수 선언 후 변수에 값을 할당

2. 연산자

- 사칙연산(+, -, *, /)
- 비교 연산자(==, !=, >, <, >=)
- 논리 연산자(!, ||, &&)
- 복합 대입 연산자
- 증감 연산자
- 자료형 확인 연산자(typeof : 해당 변수의 자료형을 추출)
- 강제 자료형 변환(Number(), String(), Boolean())