javascript

Javascript의 기본 표기

❖ JavaScript를 HTML 파일 안에 집어 넣기

```
<script type="text/javascript">
Javascript 코드
</script>
```

Javascript의 기본 표기

❖ hello.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>자바스크립트 마스터 북</title>
</head>
<body>
<script type="text/javascript">
// window.alert은 지정된 문자열을 대화 상자로 표시하기 위한 명령
window.alert('안녕하세요, 자바 스크립트!');
</script>
</body>
</html>
```

Javascript의 기본 표기

❖ <script>의 위치

실행 결과를 문서 내에 포함시키고 싶다
→ <body> 요소 내에 기술

```
<!DOCTYPE html>
<html>

<head>
<meta charset="UTF-8" />
<title>JavaScript...</title>
</head>
<body>
  ...임의의 콘텐츠...
<script type="text/javascript">

  ...JavaScript의 코드...

</script>
  ...임의의 콘텐츠...
</body>
</html>
```

웹 페이지를 조작하는 코드를
정의하고 싶다
→ </body> 요소 직전에 기술

```
<!DOCTYPE html>
<html>

<head>
<meta charset="UTF-8" />
<title>JavaScript...</title>
</head>
<body>
  ...임의의 콘텐츠...

<script type="text/javascript">

  ...JavaScript의 코드...

</script>
</body>
</html>
```

<body> 요소 안에서 직접 호출하기
위한 함수를 정의하고 싶다
→ <head> 요소 안에 기술

```
<!DOCTYPE html>
<html>

<head>
<meta charset="UTF-8" />
<title>JavaScript...</title>
<script type="text/javascript">

  ...JavaScript의 코드...

</script>
</head>
<body>
  ...임의의 콘텐츠...
</body>
</html>
```

Javascript의 기본 표기

❖ 외부 스크립트 임포트하기

- 별도의 외부 파일로 정의하고 임포트하여 사용

```
<script type="text/javascript" src="path" [charset="encoding"]>
</script>
```

- path: 스크립트 파일의 경로
- encoding: 스크립트 파일에 사용하는 문자 코드

Javascript의 기본 표기

❖ hello_ex.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>자바스크립트 마스터 북</title>
</head>
<body>
<script type="text/javascript" src="scripts/hello_ex.js"></script>
<noscript>JavaScript를 이용할 수 없습니다.</noscript>
</body>
</html>
```

❖ scripts/hello_ex.js

```
// window.alert은, 지정된 문자열을 대화 상자 표시를 위한 명령이다.
window.alert('안녕하세요, 자바 스크립트!');
```

Javascript의 기본 표기

❖ Anchor 태그에 스크립트 집어 넣기

- JavaScript 의사 프로토콜

```
<a href="javascript:스크립트코드">링크 텍스트</a>
```

```
<body>
<a href="javascript:window.alert('안녕하세요, 자바 스크립트!');">
  대화 상자를 표시</a>
</body>
```

Javascript의 기본 표기

❖ 문장의 규칙

- 문장의 맨 끝에 ;을 붙인다(옵션)
- 문장 중간에 공백이나 개행, 탭을 포함할 수 있다

```
window.
   alert
     ('안녕하세요, 자바스크립트!');
```

- 대소문자를 구분함

Javascript의 기본 표기

❖ 주석

- //
  - 한 줄 주석
- /* */
  - 복수행 주석
- /** */
  - 문서화 주석

변수와 상수

❖ var로 변수 선언하기

```
var 변수명 [=초기값], ...;
```

```
var msg;

var x;
var y;
```

```
var msg;

var x, y;
```

```
var msg = '안녕하세요! javascritp';

var x, y;
```

변수와 상수

❖ let으로 변수 선언하기

```
let 변수명 [=초기값], ...;
```

```
let msg;

let x, y;
```

- 변수의 중복을 허용하지 않음

```
let msg = 'hello';
let msg = 'world';        // 에러
```

- 블록 스코프를 가짐
  - var는 함수 스코프를 가짐

변수와 상수

❖ 식별자 명명규칙

- 변수, 함수명 : CamelCase
- 클래스명 : Pascal Case

변수와 상수

❖ 상수 선언

```
const 상수명 = 값;
```

```
const TAX = 1.08;

var price = 100;

console.log(price*TAX);
```

데이터형

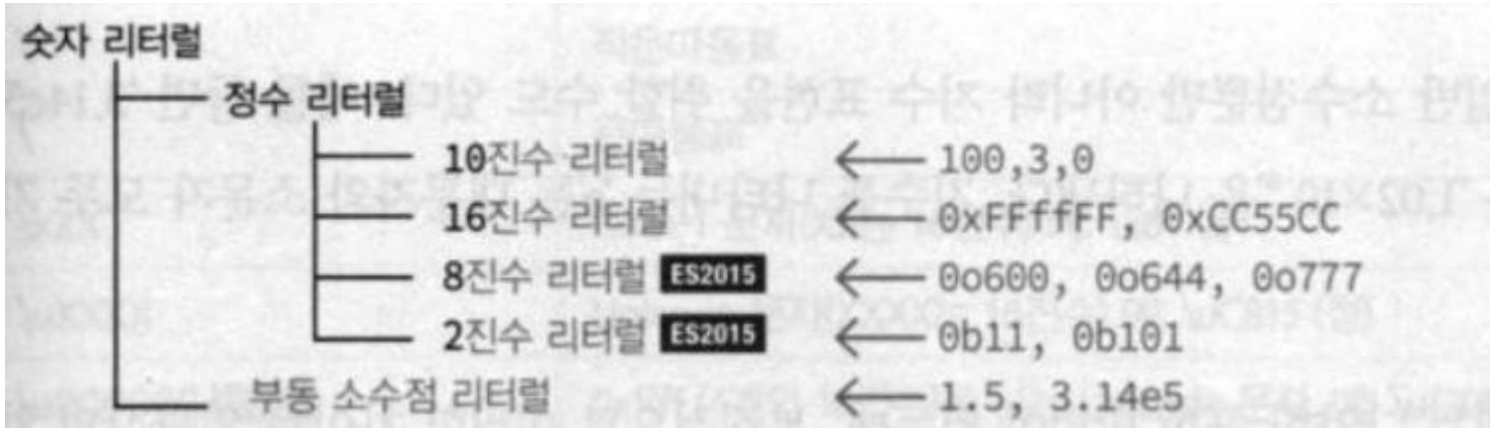
❖ javascript 데이터형

분류	데이터형	개요
기본형	숫자형(number)	$\pm 4.94065645841246544 \times 10^{-324} \sim \pm 1.79769313486231570 \times 10^{308}$
	문자열형(string)	작은따옴표/큰따옴표로 감싸인 0개 이상의 문자 집합
	논리형(boolean)	true(참)/false(거짓)
	심벌형(symbol) ES2015	심벌을 나타냄(3.2.3절을 참조)
	특수형(null/undefined)	값이 미정의된 것을 나타냄
참조형	배열(array)	데이터의 집합(각 요소에는 인덱스 번호로 접근 가능)
	객체(object)	데이터의 집합(각 요소에는 이름으로 접근 가능)
	함수(function)	일련의 처리(절차)의 집합(제4장 참조)

데이터형

❖ 리터럴

- 숫자 리터럴



- 문자 리터럴

```
'안녕하세요, JavaScript ! '
"안녕하세요, JavaScript ! "
```

데이터형

❖ 템플릿 문자열

- ` (백쿼트)로 감싼 문자열
- 문자열 안에 변수 삽입
- 복수행으로 문자열 정의

```
var name = '하은';
var str = '안녕하세요,' + name + '씨.\n오늘도 좋은 날씨네요!';
console.log(str);
```

```
var name = '하은';
var str = `안녕하세요, ${name}씨.
오늘도 좋은 날씨네요!`;
console.log(str);
```

데이터형

❖ 배열 리터럴

```
[요소1, 요소2, ... ]
```

- 배열의 요소 접근
  - 배열명[인덱스 번호]
  - 배열명[인덱스 번호][인덱스 번호]

```
var data = ['JavaScript', 'Ajax', 'ASP.NET'];
console.log(data[0]);
```

```
var data = ['JavaScript', ['jQuery', 'prototype.js'], 'ASP.NET'];
console.log(data[1][0]);
```

데이터형

❖ 객체 리터럴

```
{
   키1: 값1,
   키2: 값2,
   ...
}
```

- 객체 프로퍼티 접근
  - 객체변수.프로퍼티명
  - 객체변수['프로퍼티명']

```
var obj = { x:1, y:2, z:3 };
console.log(obj.x);
console.log(obj['x']);
```

데이터형

❖ 미정의값

- undefined
  - 변수가 선언되고 초기화하지 않은 경우
  - 객체의 미정의 프로퍼티를 참조한 경우
  - 함수에서 값을 반환하지 않은 경우

```
var x;
var obj = { a:12345 };
console.log(x);
console.log(obj.b);
```

데이터형

❖ **null**

- 아무것도 참조하지 않음을 나타내는 특수 값

- 참조형 변수의 초기값으로 주로 사용

연산자

❖ 산술 연산자

연산자	개요	예
+	숫자의 덧셈	3 + 5 // 8
-	숫자의 뺄셈	10 - 7 // 3
*	숫자의 곱셈	3 * 5 // 15
/	숫자의 나눗셈	10 / 5 // 2
%	숫자의 나머지 연산	10 % 4 // 2
++	전치 덧셈	x = 3; a = ++x; // a는 4
++	후치 덧셈	x = 3; a = x++; // a는 3
--	전치 뺄셈	x = 3; a = --x; // a는 2
--	후치 뺄셈	x = 3; a = x--; // a는 3

연산자

❖ 증가, 감소 연산자(++, --)

```
var x = 3;
var y = x++;
console.log(x);  // 4
console.log(y);  // 3

var x = 3;
var y = ++x;
console.log(x);  // 4
console.log(y);  // 4
```

연산자

❖ 소수점 연산

```
console.log(0.2*3)        // 0.6000000000000001
```

- 소수를 2진수로 표현하기 때문에 오차 발생
- 동등연산시(==) 주의

```
console.log(0.2 * 3)                          // 0.6000000000000001
console.log(0.2 * 3 === 0.6);                 // false

console.log(((0.2 * 10) * 3) / 10);           // 0.6
console.log((0.2 * 10) * 3 === 0.6 * 10);     // True
```

연산자

❖ 대입 연산자

연산자	개요	예
=	변수 등에 값을 대입한다.	x = 1
+=	좌변값에 우변값을 더한 값을 대입한다	x = 3; x += 2 // 5
-=	좌변값에 우변값을 뺀 값을 대입한다	x = 3; x -= 2 // 1
*=	좌변값에 우변값을 곱한 값을 대입한다	x = 3; x *= 2 // 6
/=	좌변값에 우변값을 나눈 값을 대입한다	x = 3; x /= 2 // 1.5
%=	좌변값에 우변값을 나눈 나머지 값을 대입한다	x = 3; x %= 2 // 1
&=	좌변값에 우변값을 비트 AND 연산한 값을 대입한다	x = 10; x &= 5 // 0
|=	좌변값에 우변값을 비트 OR 연산한 값을 대입한다	x = 10; x |= 5 // 15
^=	좌변값에 우변값을 비트 XOR 연산한 값을 대입한다	x = 10; x ^= 5 // 15
<<=	좌변값에 우변값만큼 좌측 시프트한 결과를 대입한다	x = 10; x <<= 1 // 20
>>=	좌변값에 우변값만큼 우측 시프트한 결과를 대입한다	x = 10; x >>= 1 // 5
>>>=	좌변값에 우변값만큼 우측 시프트한 결과를 대입한다 (* Unsigned)	x = 10; x >>>= 2 // 2

연산자

❖ 기본형과 참조형에 따른 대입의 차이

- 기본형의 대입 : 값의 복사
- 참조형의 대입 : 원본(인스턴스)의 공유

```
var x = 1;
var y = x;
x = 2;
console.log(y);           // 1

var data1 = [0, 1, 2];
var data2 = data1;
data1[0] = 5;
console.log(data2);       // [5, 1, 2]
```

```
var data1 = ['JavaScript', 'Ajax', 'ASP.NET'];
var data2 = ['JavaScript', 'Ajax', 'ASP.NET'];
console.log(data1 == data2);      // false
```

연산자

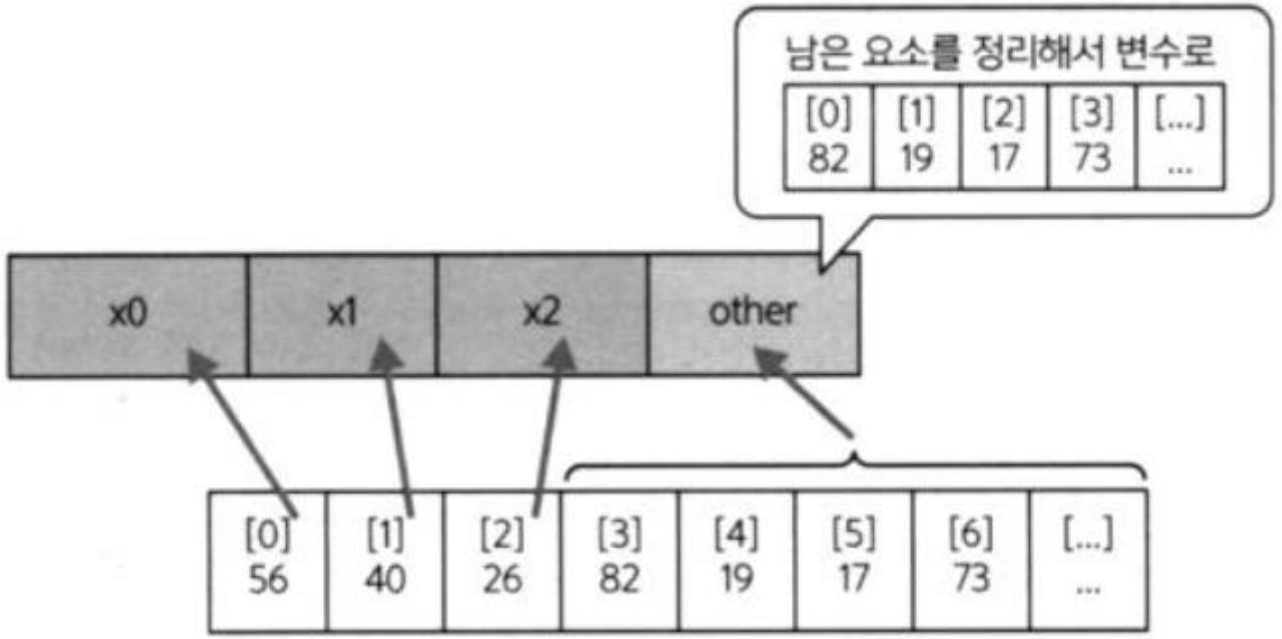
❖ 분할 대입(destructuring assignment) - 배열

```
let data = [56, 40, 26, 82, 19, 17, 73, 99];
let [x0, x1, x2, x3, x4, x5, x6, x7] = data

console.log(x0);
console.log(x1);
console.log(x2);
console.log(x3);
console.log(x4);
console.log(x5);
console.log(x6);
console.log(x7);
```

연산자

❖ 나머지 연산자(...)

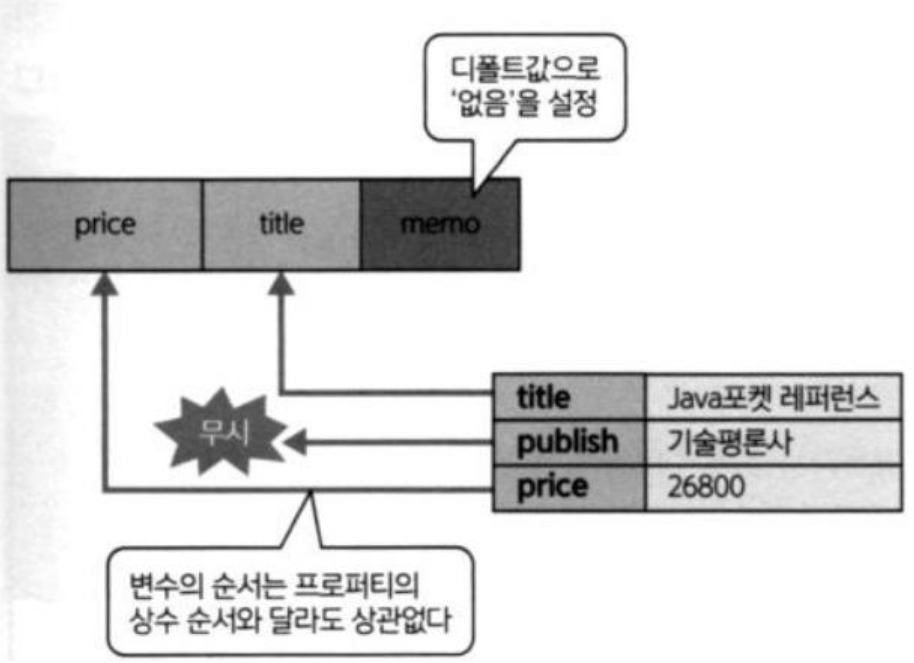
```
let data = [56, 40, 26, 82, 19, 17, 73, 99];
let [x0, x1, x2, ...other] = data

console.log(x0);
console.log(x1);
console.log(x2);
console.log(other);       // [82, 19, 17, 73, 99]
```



연산자

❖ 분할 대입 - 객체

```
let book = {
   title: 'Java포켓 레퍼런스',
   publish: '기술평론사',
   price: 2680
};
let { price, title, memo = '없음' } = book;

console.log(title);       // Java포켓 레퍼런스
console.log(price);       // 2680
console.log(memo);        // 없음
```



연산자

❖ 중첩된 객체 분해하기

```
let book = {
  title: 'Java포켓 레퍼런스 ',
  publish: '기술평론사',
  price: 26800,
  other: {
    keywd: 'Java SE 8',
    logo: 'logo.jpg'
  }
};

let { title, other, other: { keywd } } = book;

console.log(title);
console.log(other);
console.log(keywd);
```

연산자

❖ 변수의 별칭 지정

```
let book = {
  title: 'Java포켓 레퍼런스',
  publish: '기술평론사'
};
let { title: name, publish: company } = book;

console.log(name);
console.log(company);
```

연산자

❖ 비교 연산자

연산자	개요	예
==	좌변과 우변의 값이 같을 경우, true	5 == 5 // true
!=	좌변과 우변의 값이 같지 않을 경우, true	5 != 5 // false
<	좌변이 우변보다 작은 경우, true	5 < 5 // false
<=	좌변이 우변보다 작거나 같을 경우, true	5 <= 5 // true
>	좌변이 우변보다 클 경우, true	5 > 3 // true
>=	좌변이 우변보다 크거나 같을 경우, true	5 >= 3 // true
===	좌변과 우변의 값이 같고 데이터형도 같은 경우, true	5 === 5 // true
!==	좌변과 우변의 값이 같지 않을 경우 또는 데이터형이 다른 경우, true	5 !== 5 // false
?:	'조건식? 식1: 식2'. 조건식이 true인 경우는 식1을, false인 경우는 식2를 취한다	(x==1) ? 1: 0 // 1 또는 0

연산자

❖ 등가 연산자

좌변과 우변의 형	데이터형	평가 기준
동일	문자열/수치/논리형	단순히 쌍방의 값이 동일한지를 판단
	배열/객체	참조 장소가 동일한지를 판단
	null/undefined	쌍방이 모두 null/undefined, 또는 null과 undefined의 비교는 모두 true
다름	문자열/수치/논리형	문자열/논리형을 수치로 변환한 후에 판단
	객체	기본형으로 변환한 후에 판단

연산자

❖ 등가 연산자와 동치 연산자

```
console.log('3.14E2' == 314);      // true
console.log('0x10' == 16);         // true
console.log('1' == 1);             // true
```

```
console.log('3.14E2' === 314);     // false
console.log('0x10' === 16);        // false
console.log('1' === 1);            // false
```

연산자

❖ 조건 연산자

```
var x = 80;
console.log((x >= 70) ? '합격' : '불합격');
```

연산자

❖ 논리 연산자

연산자	개요	예
&&	좌우식이 모두 true인 경우, true	100 === 100 && 1000 === 1000 // true
‖	좌우식의 어느 쪽이든 true인 경우, true	100 === 100 ‖ 1000 === 500 // true
!	식이 false인 경우, true	!(10 > 100) // true

```
var x = 1;

if (x === 1) { console.log('안녕하세요'); }
x ===1 && console.log('안녕하세요');
```

```
var msg = '';
msg = msg || '안녕하세요, 자바스크립트!';
console.log(msg);
```

연산자

❖ 그외의 연산자

연산자	개요
,(콤마)	좌우의 식을 계속해서 실행(2.5.5절 참조)
delete	객체의 프로퍼티나 배열의 요소를 삭제
instanceof	객체가 지정된 클래스의 인스턴스인지를 판정(5.3.3절 참조)
new	새로운 인스턴스를 생성(3.1.2절 참조)
typeof	오퍼랜드의 데이터형을 취득
void	미정의 값을 되돌림(6.5.3절 참조)