6장

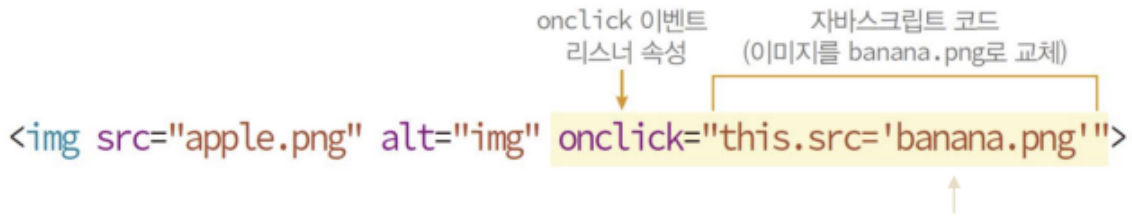
자바스크립트 언어

- 자바스크립트 언어란 : 웹 프로그래밍 개념 창시

→ html 문서에 내장 , 스크립트 언어, 단순

- 역할 : 사용자의 입력 및 계산 , 웹 페이지 내용 및 모양의 동적 제어 , 브라우저 제어 , 웹 서버와의 통신 , 웹 애플리케이션 작성
- 위치 :

1. html 태그의 이벤트 리스너 속성에 작성



2. <script></script> 태그에 작성

→ <head></head> 나 <body></body> 내 어디든 가능

→ 웹 페이지 내에 여러 번 삽입 가능

3. 자바스크립트 파일에 작성

→ 확장자 .js 파일에 저장

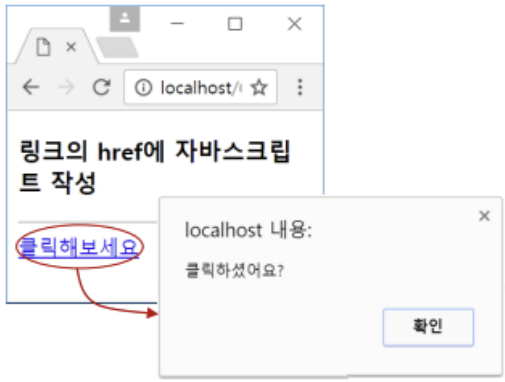
→ <sript> 태그 없이 자바스크립트 코드만 저장

→ 여러 웹 페이지에서 불러 사용 가능

```
<script src="파일이름.js">
  // HTML5부터 이곳에 자바스크립트 코드 추가 작성하면 안 됨
</script>
```

4. url 부분에 작성

```
<!DOCTYPE html>
<html>
<head><title>URL에 자바스크립트 작성</title>
</head>
<body>
<h3>링크의 href에 자바스크립트 작성</h3>
<hr>
<a href="javascript:alert('클릭하셨어요?')">
  클릭해보세요</a>
</body>
</html>
```



- 자바스크립트로 HTML 콘텐츠로 출력

→ documnet.write() : 브라우저 화면에 출력

→ document.writeln() : \n 을 덧붙여 출력 → 빈칸 하나 출력 하는 것

- 자바스크립트 다이얼로그

→ 프롬프트 다이얼로그 : prompt(“메세지” , “디폴트 입력값”) 함수 → 사용자로부터 문자열을 입력 받아 리턴

→ 확인 다이얼로그 : confirm(”메세지”) 함수 → “메세지”를 출력하고 확인 / 취소 버튼을 가진 다이얼로그 출력

→ 경고 다이얼로그 : alert(”메세지”) 함수 → “메세지”와 “확인” 버튼을 가진 다이얼로그 출력, 메세지 전달

- 자바스크립트 식별자

: 자바스크립트 프로그램의 변수, 상수, 함수의 이름

<규칙>

 첫 번째 문자 : 알파벳(A-Z, a-z), 언더스코어(), *$ 문자만 사용 가능*
 두 번째 이상 문자 : 알파벳, 언더스코어(), 0-9, $ 사용 가능
 대소문자는 구분되어 다루어짐
 myHome과 myhome은 다른 식별자
 자바스크립트 예약어 사용 불가
 false, for, if, null 등 자바스크립트 예약어 사용 불가

식별자 사용 사례

```
6variable;        // (x) 숫자로 시작할 수 없음
student_ID;       // (0)
_code;            // (0) 맞지만 권하지 않음
if;               // (x) 예약어 if 사용 불가
%calc             // (x) % 사용 불가
bar, Bar;         // (0) bar와 Bar는 서로 다른 식별자임에 주의
```

- 자바스크립트 문장

: 자바 스크립트의 기본 단위

: 문장과 문장을 구분하기 위해 세미콜론 (;) 사용

→ 주석문

// : 한 라인 주석

/* ~ */ : 여러 라인 주석

- 데이터 타입

⇒ 숫자 타입, 논리 타입, 문자열 타입, 객체 레퍼런스 타입, null

*자바스크립트에는 문자 타입 없음. 문자열로 표현

- 변수

: 자바스크립트 데이터 저장 공간

→ 변수 선언 : 변수 이름을 정하고, 저장 공간 할당

var 키워드 , var 없이도 가능

→ 변수 타입 없음!! ; 즉, 변수에 저장되는 값에 대한 제약이 없음!!

⇒ 지역변수 & 전역변수

지역변수	전역변수
함수 내에 var 키워드로 선언	함수 밖에 선언되거나, 함수 내에 var 키워드 없이 선언된 변수
선언된 함수 내에서만 사용	프로그램 전역에서 사용

```
var x;       // 전역변수 x
function f() {
   var y;    // 지역변수 y 선언
   x = 10;   // 전역 변수 x에 10 저장
   y = 10;   // 지역 변수 y에 10 저장
   z = 10;   // 새로운 전역변수 z 선언. 10으로 초기화
}
```

→ this 로 전역변수 접근

: this.전역변수

```
var x;       // 전역변수
function f() {
   var x;    // 지역변수
   x = 1;    // 지역변수 x에 1 저장
   this.x = 100; // 전역변수 x에 100 저장
}
```

- 상수

: 데이터 값 그 자체

상수의 종류		특징	예
정수	8진수	0으로 시작	var n = 015; // 8진수 15. 10진수로 13
	10진수		var n = 15; // 10진수 15
	16진수	0x로 시작	var n = 0x15; // 16 진수 15. 10진수로 21
실수	소수형		var height = 0.1234;
	지수형		var height = 1234E-4; // $1234x10^{-4}$ = 0.1234
논리	참	true	var condition = true;
	거짓	false	var condition = false;
문자열		""로 묶음	var hello = "안녕하세요";
		''로 묶음	var name = 'kitae';
기타	null	값이 없음을 뜻함	var ret = null;
	NaN	수가 아님을 뜻함	var n = parseInt("abc"); // 이때 parseInt()는 NaN을 리턴

⇒ 문자열 상수

: “ “ 와 ‘ ‘ 모두 사용

문자열 내 문자열

```
<p onmouseover = "document.body.style.color = 'brown' ">
```

→ 문자를 그대로 사용하고자 하는 경우 \로 사용할 것.

- 식과 연산

연산 종류	연산자	연산 종류	연산자
산술	+ - * / %	대입	= *= /= += -= &= ^= |= <<= >>= >>>=
증감	++ --	비교	> < >= <= == !=
비트	& | ^ ~	논리	&& || !
시프트	>> << >>>	조건	? :

⇒ 산술 연산자 : + , - , * , / % =⇒ 연산의 결과는 항상 실수

⇒ 증감 연산자 : ++ , - -

연산자	내용	연산자	내용
a++	a를 1 증가하고 증가 전의 값 반환	++a	a를 1 증가하고 증가된 값 반환
a--	a를 1 감소하고 감소 전의 값 반환	--a	a를 1 감소하고 감소된 값 반환

차례로 후위 연산자와 전위 연산자

⇒ 대입 연산자 : 오른쪽 식의 결과를 왼쪽 변수에 대입

연산자	내용	연산자	내용
a = b	b 값을 a에 대입	a &= b	a = a & b와 동일
a += b	a = a + b와 동일	a ^= b	a = a ^ b와 동일
a -= b	a = a - b와 동일	a |= b	a = a | b와 동일
a *= b	a = a * b와 동일	a <<= b	a = a << b와 동일
a /= b	a = a / b와 동일	a >>= b	a = a >> b와 동일
a %= b	a = a % b와 동일	a >>>= b	a = a >>> b와 동일

⇒ 비교 연산자 : 두 값 비교, true 나 false의 결과를 내는 연

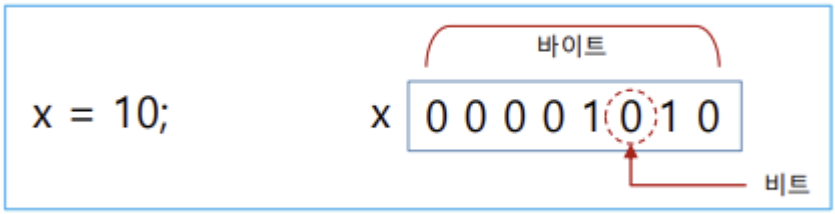
연산자	내용	연산자	내용
a < b	a가 b보다 작으면 true	a >= b	a가 b보다 크거나 같으면 true
a > b	a가 b보다 크면 true	a == b	a가 b와 같으면 true
a <= b	a가 b보다 작거나 같으면 true	a != b	a가 b와 같지 않으면 true

⇒ 논리 연산자 : and, or, not

연산자	별칭	내용
a && b	논리 AND 연산	a, b 모두 true일 때 true 리턴
a || b	논리 OR 연산	a, b 중 하나라도 true이면 true 리턴
!a	논리 NOT 연산	a가 true이면 false 값을, false이면 true 값 리턴

⇒ 조건 연산자 : condition ? expT : expF

⇒ 비트 연산

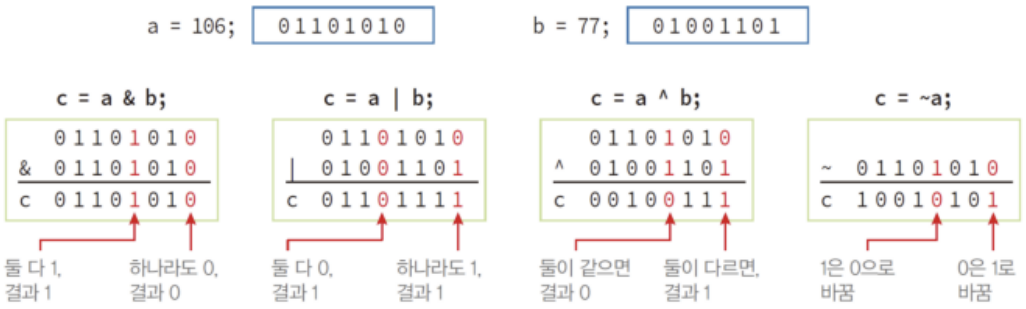


→ 비트 연산 종류

1.비트 논리 연산

□ 비트 논리 연산

연산자	별칭	연산 설명
a & b	비트 AND 연산	두 비트 모두 1이면 1, 그렇지 않으면 0
a | b	비트 OR 연산	두 비트 모두 0이면 0, 그렇지 않으면 1
a ^ b	비트 XOR 연산	두 비트가 다르면 1, 같으면 0
~ a	비트NOT 연산	1을 0으로, 0을 1로 변환



2. 비트 시프트 연산

→ 시프트 : 저장 공간에서 비트들의 오른쪽 / 왼쪽 이동

연산자	별칭	설명
a << b	산술적 왼쪽 시프트	a의 비트들을 왼쪽으로 b번 이동. 최하위 비트의 빈자리는 0으로 채움. 한 비트 시프트마다 곱하기 2의 효과 발생. a 값은 변화 없음
a >> b	산술적 오른쪽 시프트	a의 비트들을 오른쪽으로 b번 이동. 최상위 비트의 빈자리는 시프트 전 최상위비트로 채움. 한 비트 시프트마다 나누기 2의 효과 발생. a 값은 변화 없음
a >>> b	논리적 오른쪽 시프트	a의 비트들을 오른쪽으로 b번 이동. 최상위 비트의 빈자리는 0으로 채움. a 값은 변화 없음

→ 실제 활용 할 때는 숫자를 8비트 2진수로 변환하는 과정이 필요하다.

```
function digit8(v) { // 숫자 v를 8비트 2진수로 변환
    var str="";
    for(i=0; i<8; i++, v<<=1) {
        if((v & 0x80)) str += "1";
        else str += "0";
    }
    return str;
}
```

⇒ 문자열 연산자

: 문자열 연결 → + , +=

: 문자열 비교 → ≠ , == , > , < , ≤ , ≥ (사전 순으로 비교 결과 리턴)

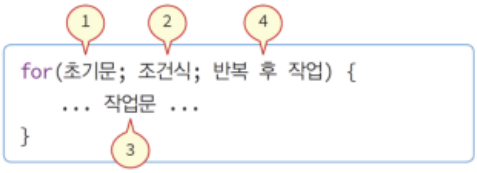
- if , if - else 문
- switch 문 : 값이 서로 다른 코드를 실행할 때! → case 문 + break 문

→ case 문의 '값' : 상수(리터럴)만 가능 (변수나 식은 사용 불가)

→ break 문 : switch 문 종료

- 반복문

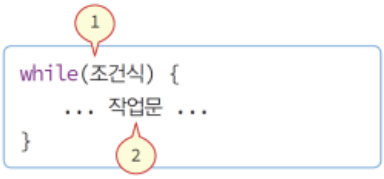
□ for 문




```
// 0에서 9까지 출력
for(var i=0; i<10; i++) {
   document.write(i);
}
```

```
0123456789
```

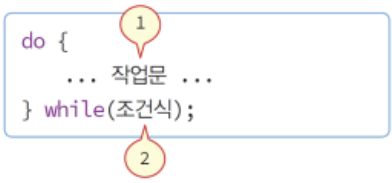
□ while 문




```
var i=0;
while(i<10) {  // i가 0에서 9까지 출력
   document.write(i);
   i++;
}
```

```
0123456789
```

□ do-while 문




```
var i=0;
do {  // i가 0에서 9까지 출력
   document.write(i);
   i++;
} while(i<10);
```

```
0123456789
```

*문자열을 숫자로 바꾸는 함수 : parseInt (n);

→ break 문 : 가장 안쪽 반복문 하나만 벗어나도록 제어

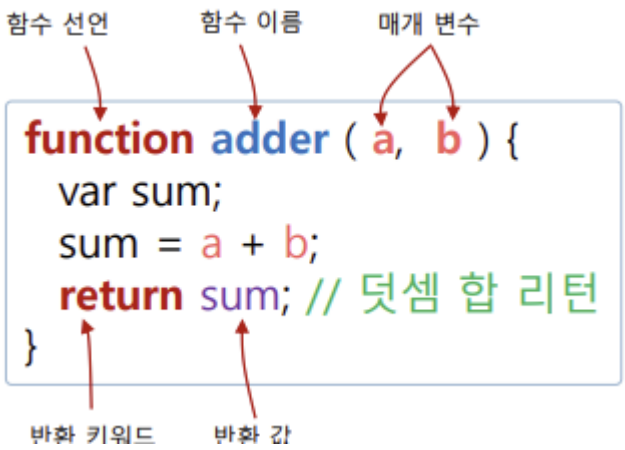
→ continue 문 : 반복 코드 실행 중단, 다음 반복으로 점프

- 함수

: 목적을 가지고 작성된 코드 블록

: 데이터 전달 받아 처리한 후 결과를 돌려주는 코드 블록

⇒ 구성



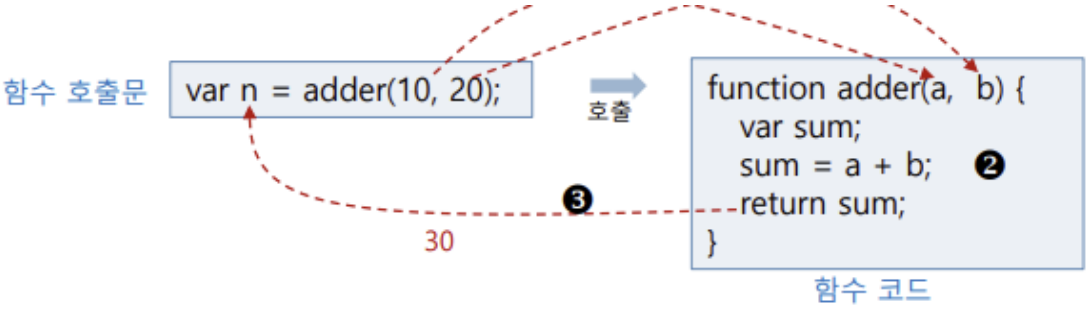
function 함수이름 (arg1, arg2 ,……, arg n) {
프로그램 코드

결과를 리턴하는 return 문

}

⇒ 호출

: 함수의 코드 실행 요청



⇒ 자바 스크립트에서 제공하는 전역 함수

1. eval() 함수 : 자바 스크립트 식을 계산 하고 결과 리턴
2. parseInt() 함수 : 주어진 문자열을 10진 정수로 바꾸어 리턴
3. isNaN() 함수 : 주어진 값이 숫자가 아니면 true 리

전역 함수명	설명
eval(exp)	exp의 자바스크립트 식을 계산하고 결과 리턴
parseInt(str)	str 문자열을 10진 정수로 변환하여 리턴
parseInt(str, radix)	str 문자열을 radix 진수로 해석하고, 10진 정수로 바꾸어 리턴
parseFloat(str)	str 문자열을 실수로 바꾸어 리턴
isFinite(value)	value가 숫자이면 true 리턴
isNaN(value)	value가 숫자가 아니면 true 리턴