

HTML – 08

JavaScript 기초

자바스크립트 소개

- 자바스크립트(JavaScript): 웹 페이지를 구성하는 3가지 언어 중 동작을 프로그래밍하는 언어
- 웹의 표준 프로그래밍 언어
- 모든 웹브라우저들은 자바스크립트를 지원



자바스크립트 역사

- 넷스케이프의 브랜든 아이크(Brendan Eich)가 개발
- 처음에는 라이브스크립트(LiveScript)- sun의 자바가 유행 -java script
- 자바스크립트가 잘 되자, MS에서 Jscript라는 언어를 개발해 IE에 탑재하였는데, 이 두 스크립트가 너무 제각각이라, 표준이 필요하게 되었다.
- 표준을 위해 자바스크립트를 **ECMA(European Computer Manufacturers Association)** 라는 정보와 통신시스템의 비영리 표준 기구에 제출하였고 *표준에 대한 작업을 ECMA-262란 이름으로 1996년 11월에 시작해 1997년 6월에 채택되었다.*
- 오리지널 자바스크립트 ES1 ES2 ES3 (1997 - 1999)
- 첫번째 주요 개정판 ES5(2009)
- 두번째 개정판 ES6(2015)
- 2016년부터 새 버전의 이름은 연도별로 지정된다.






자바스크립트 특징

- 인터프리트 언어
 - 컴파일 과정을 거치지 않고 바로 실행시킬 수 있는 언어
- 동적 타이핑(dynamic typing)
 - 변수의 자료형을 선언하지 않고도 변수를 사용할 수 있는 특징
- 구조적 프로그래밍 지원
 - C언어의 구조적 프로그래밍 지원.
즉 if else, while, for 등의 제어 구조를 완벽 지원
- 객체 기반
 - 객체 기반 언어로서 내장 객체를 지원.
자바스크립트의 객체는 연관 배열(associative arrays)
- 함수형 프로그래밍 지원
 - 자바스크립트 함수는 일급 객체(first-class object).
- 프로토타입 기반
 - 상속을 위해 클래스 개념 대신 프로토타입 사용

자바스크립트의 용도

- 웹 페이지에 기능을 더하여 동적인 화면을 구성
- HTML 페이지 변경 및 요소와 콘텐츠의 추가, 제거
- CSS를 이용한 요소 스타일 변경
- 마우스, 키보드 이벤트에 대한 스크립트 실행
- 사용자 입력 값에 대한 검증 작업
- AJAX기술을 이용한 웹 서버와의 통신

자바스크립트의 미래

- 본래 클라이언트 웹 페이지를 위한 프로그래밍 언어였지만 그 용도는 점점 더 확장되고 있다.
 - Node.js : 웹 서버와 같은 애플리케이션을 작성하기 위해 설계된 서버-사이드(Server-Side) 소프트웨어 시스템 
 - jQuery : 자바스크립트 라이브러리 
 - JSON : 자바스크립트 객체 표기법(JavaScript Object Notation) 자바스크립트의 리터럴 표현식을 활용한 간단한 데이터 형식 포맷
특정 언어에 종속되지 않는 독립적인 데이터 형식으로 XML을 대체하는 경량의 데이터 교환 형식. 

JSON
JavaScript Object Notation

자바스크립트 구문 및 주석

- 구문(syntax) : 프로그램이 구성되는 규칙의 집합
 - 유형 - 고정 값(리터럴), 변수 값(변수)
 - 문자는 쌍 따옴표(“)나, 홑 따옴표(‘)로 감싸서 표현
 - 변수는 var, let, const 키워드를 사용하여 선언
 - 식별자는 숫자로 시작할 수 없고, 대소문자를 구분

- // - 단일 문장 주석

```
// id가 heading1인 헤딩요소를 찾아서 내용을 바꾼다.  
document.getElementById("heading1").innerHTML = "My HomePage";
```

- /* */ - 다중 문장 주석

```
/*  
    이 코드는 웹 페이지의 헤딩의 내용을 변경한다.  
*/  
document.getElementById("heading1").innerHTML = "My HomePage";
```

자바스크립트 출력

- 출력 속성 및 메소드
 - innerHTML - 출력 내용속에 html태그가 포함, html태그실행
 - innerText - 일반 텍스트문자로 출력, html태그도 문자로 출력
 - document.write() - 로딩 시 웹 페이지에 데이터 출력. 테스트 용
 - window.alert() - 별도의 대화상자를 띄워 데이터 출력
 - console.log() - 브라우저 콘솔을 통해 데이터 출력. 디버깅 용
- <script> : 사용자 측 스크립트를 포함하는데 사용되는 태그
 - src 속성을 통해 외부 스크립트 파일을 포함시켜 사용한다.
 - 일반적으로는 스타일 조작, 입력 양식의 유효성 검사 등 콘텐츠의 동적인 변경을 위해 사용된다.

자바스크립트의 위치(1/3)

- 내부 자바스크립트 - <head>, <body> 양쪽 배치 가능

```
<!DOCTYPE HTML>
<html>
<head>
  <title>My JavaScript </title>
  <script>
    document.write("Hello World!");
  </script>
</head>
<body></body>
</html>
```

```
<!DOCTYPE HTML>
<html>
<head>
  <title>My JavaScript </title>
</head>
<body>
  <script>
    document.write("Hello World!");
  </script>
</body>
</html>
```

자바스크립트의 위치(2/3)

- 외부 자바스크립트 - <head>, <body> 양쪽 배치 가능
- 장점
 - HTML과의 코드 분리로 유지보수에 용이하며 가독성이 높아짐
 - 캐시된 JavaScript파일로 인해 페이지 로드 속도가 빨라짐

```
<html>
<head>
  <script src="commonScript.js"></script>
</head>
<body>
  <script src="myScript.js"></script>
</body>
</html>
```

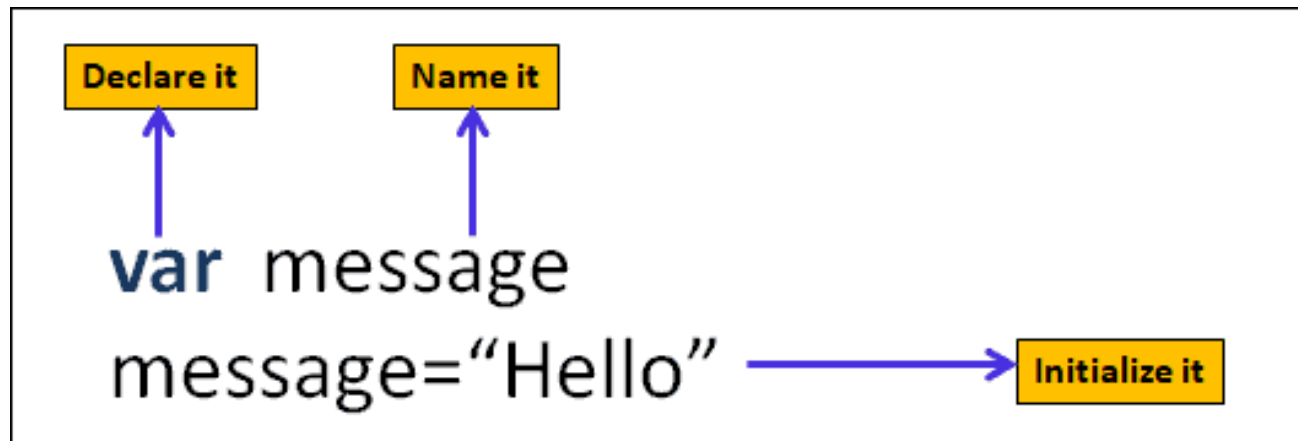
자바스크립트의 위치(3/3)

- 인라인 자바스크립트

```
<!DOCTYPE html>
<html>
<body>
  <button type="button" onclick="alert('Hi!! ')">버튼을
  누르세요!</button>
</body>
</html>
```

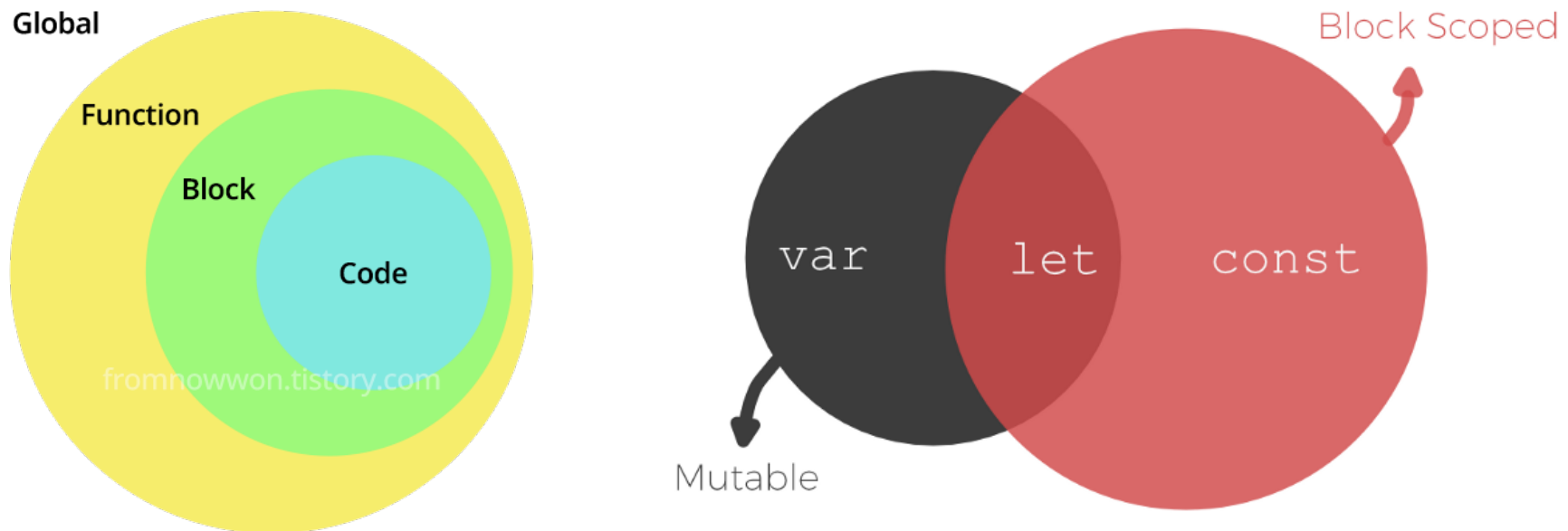
변수와 변수 키워드

- **변수(variable)**는 데이터를 저장할 수 있는 공간으로 값이 변경될 수 있으며 **변수 키워드**를 사용해서 선언한다.
 - 변수의 선언(declare) : 자바스크립트는 변수 키워드를 사용해서 변수를 선언한다.
 - 선언되지 않은 변수에 접근 시 오류가 발생한다.
 - 변수의 이름(name) : 식별 가능한 식별자(identifier)
 - 변수의 초기화(initialize) : 사용 전 초기 값을 저장한다.
 - 데이터를 저장하기 위해 할당 연산자(=)를 사용한다.



변수와 변수 키워드

- 변수 키워드(variable keyword)를 사용해서 변수를 선언(declare)한다.
 - var : 함수 범위에서 유효하며 재선언과 재할당이 가능하다.
 - let : 블록 범위에서 유효하며 재선언은 불가, 재할당만 가능하다.
 - const : 블록 범위에서 유효한 상수 선언 키워드로 재선언과 재할당이 불가하다.



변수 명명 규칙

- 꼭 지켜야 하는 룰
 - 식별자는 영문자, 언더스코어(_), 달러(\$)로 시작해야 한다.
 - 첫 자는 숫자로 시작할 수 없으며, 두번째 글자부터 가능하다.
 - 대소문자를 구별하므로 'javascript'와 'javaScript'는 다른 식별자다.
 - 예약어(자바스크립트에서 이미 사용중인 단어)는 사용할 수 없다.
 - break, default, final, for, new, null, try, this 등등
- 지키면 좋은 룰
 - 의미 없는 이름으로 변수 명 사용하지 않기
 - let a; //어떠한 값이 저장되었는지 찾기 어렵고 활용도가 떨어짐
 - 추상적인 이름 사용하지 않기
 - let name; //조금 더 구체적인 이름으로 표기하도록 권장
 - 카멜표기법으로 표기하기
 - 띄어쓰기를 대신하여 각 단어의 첫 문자를 대문자로 표기

document.write(1)

<script>

```
var day = new Date(); //기본형식의 날짜와 시간을 반환한다.  
document.write(day + "<br>");
```

```
var now = day.toLocaleString(); //현지버전으로 날짜와 시간을 반환한다.  
document.write(now + "<br>");
```

</script>

<body>

<h1>Hello~</h1>

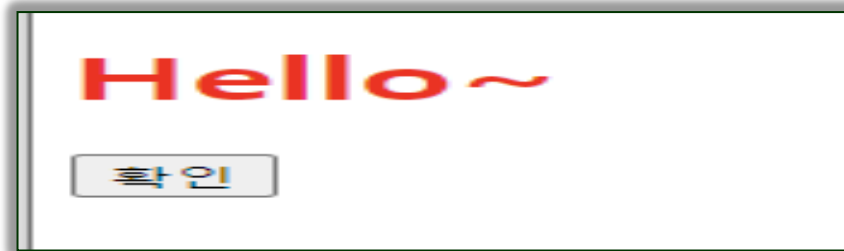
</body>

Hello~

Mon Sep 05 2022 10:40:15 GMT+0900 (한국 표준시)
2022. 9. 5. 오전 10:40:15

document.write(2)

```
<body>
  <h1>Hello~</h1>
  <input type="button" value="확인"
        onclick="proc1()">;
  <script>
  function proc1(){
```



Mon Sep 05 2022 10:55:32 GMT+0900 (한국 표준시)
2022. 9. 5. 오전 10:55:32

```
    var day = new Date(); //기본형식의 날짜와 시간을 반환한다.
    document.write(day + "<br>");
```

```
    now = day.toLocaleString(); //현지버전으로 날짜와 시간을 반환한다.
```

```
    document.write(now + "<br>");
    document.body.style.fontSize = "2.0em";
    //---→document가 새롭게 생성, 기존의 웹페이지를 덮어쓰기 효과
```

```
}
</script>
</body>
```


innerHTML/innerText(1)

<h1>Hello~</h1>

<div id="result1"></div>

- *Id=result1인 div요소에 날짜 출력*
document.getElementById('result1')
- *스크립트의 위치에 따라 window.onload= function(){ }*
이 필요

innerHTML/innerText(2)

<script>

- `var day = new Date();` //기본형식의 날짜와 시간을 반환한다.
- `var now = day.toLocaleString();` //현지버전으로 날짜와 시간을 반환한다.
- `window.onload = function(){`
 `var vres = document.getElementById('result1');`

```
var str = day + "<br>";  
str += now + "<br>";
```

```
//vres.innerText = str;  
vres.innerHTML = str;
```

```
}
```

</script>

Hello~

Mon Sep 05 2022 11:21:46 GMT+0900 (한국 표준시)
2022. 9. 5. 오전 11:21:46

Hello~

Mon Sep 05 2022 12:51:00 GMT+0900 (한국 표준시)
2022. 9. 5. 오후 12:51:00

지역 변수

- 함수 안에서 선언된 변수는 함수 안에서만 사용 가능하다.
- 때문에 다른 함수에서도 똑같은 이름으로 선언이 가능하다.
- 지역 변수는 함수가 종료되면 자동적으로 소멸된다.

```
function add(a, b) {  
  var sum = 0;  
  
  sum = a + b;  
  return sum;  
}
```

외부에서는 sum을 사용할 수 없다.

지역 변수

```
function sub (a, b){  
    var res = a - b;  
}  
window.onload = function() {  
    sub(10, 4);  
    document.write("sub=" + res); //오류발생  
}
```

함수 범위 내 유효한
지역 변수는 외부에서
사용이 불가하여
오류를 발생시킨다.

```
function sub (a, b){  
    return a - b;  
}  
window.onload = function() {  
    document.write("sub=" + sub(10, 4));  
}
```

반환 값을 이용하자.

전역 변수

- 함수 외부에서 선언된 변수
- 웹 페이지 상의 모든 스크립트와 모든 함수는 전역 변수를 사용할 수 있다.
- 전역변수는 사용자가 웹 페이지를 닫으면 소멸된다.

```
var res = 0;
```

```
function sub (a, b){  
  res = a - b;  
}
```

```
window.onload = function() {  
  sub(10, 4);  
  document.write("sub=" + res);  
}
```

전역 변수

- 선언되지 않은 변수에 값 대입 시 자동으로 전역 변수가 된다.
- 다음 예시 문장의 username 변수도 함수 안에 존재하지만 전역 변수로 선언한 것과 마찬가지이다.
- 선언되지 않은 변수는 예상치 못한 결과를 가져오며 엄격모드에서는 에러를 발생시키므로 사용을 지양한다.

```
function add(a, b) {  
    userName = "초파";  
    sum = a + b;  
}
```

```
function sub(a,b){  
    userName = "나초";  
    sum = a - b;  
}
```

```
window.onload = function() {  
    add(4,5);  
    document.write(userName);  
    document.write("add=" + sum);  
  
    sub(10, 4);  
    document.write(userName);  
    document.write("sub=" + sum);  
}
```

자바스크립트 자료형

- 원시 타입(primitive type)
 - 숫자(number) - 정수나 실수
 - 문자열(string) - 문자가 연결된 것, " " 나 " "로 표현
 - 불리언(Boolean) - true 또는 false
 - Undefined - 값이 정해지지 않은 상태
 - Null - 값이 비어있는 상태
 - 심볼(Symbol) - es6에 포함
유일성이 보장된 자료형으로 충돌 위험이 없는 객체의 유일한 프로퍼티 키를 만들기 위해 사용된다
- 객체 타입(object type)
 - 객체(object) - 데이터의 타입이 함수 또는 배열 등의 객체

연산자(1/8)

- 산술 연산자

연산자	설명	예
+	덧셈	$x = 3 + 2$
-	뺄셈	$x = 3 - 2$
*	곱셈	$x = 3 * 2$
/	나눗셈	$x = 3 / 2$
%	나머지	$x = 3 \% 2$
++	증가	$++x, x++$
--	감소	$--x, x--$
**	지수화	$x = 5 ** 2$

- $x=5**2$ Math.pow(5,2)와 동일

- 대입 연산자

- 변수에 값을 할당한다.
- '=' 는 오른쪽 값을 왼쪽 변수에 저장한다는 의미이다.
- '같다'의 의미는 '=='를 사용

연산자	의미	예
=	$x = y$	$x = y$

- 복합 대입 연산자

연산자	의미	예
+=	$x = x + y$	$x += y$
-=	$x = x - y$	$x -= y$
*=	$x = x * y$	$x *= y$
/=	$x = x / y$	$x /= y$
%=	$x = x \% y$	$x \% = y$

연산자(2/8)

- 문자열에서의 ' + ' 연산자 (연결 연산자)
 - 문자열을 결합하는 용도로도 사용된다.
 - 즉 + 연산자가 문자열에서 사용되면 문자열 결합의 의미가 된다.

```
s1 = "Welcom to ";  
s2 = "Javascript";  
s3 = s1 + s2;
```

- 숫자와 문자열을 ' + ' 연산자로 합하면 숫자를 문자열로 변환하여, 결합된 문자열을 반환한다.

```
x = 1 + 1;  
y = "Car" + 1;  
document.write(x + "<br>");  
document.write(y + "<br>");
```

연산자(3/8)

- 예제

```
<script>
  var s;
  s = 100;
  document.write(s + "<br>");

  s = "홍길동";
  document.write(s + "<br>");
</script>
```

```
<script>
  var s = "Hello World";
  var t = "How are you" + " today?";

  document.write(s + "<br>");
  document.write(t + "<br>");
  document.write(s.toUpperCase() + "<br>");
</script>
```

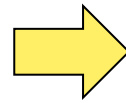
연산자(4/8)

- 예제

```
<script>
```

```
var str1 = 16 + 4 + "Hello ";
```

```
var str2 = "Hello" + 16 + 4;
```



```
20Hello  
Hello164
```

```
document.write(str1);  
document.write("<br>");  
document.write(str2);
```

```
</script>
```

덧셈 연산자는 왼쪽에서 오른쪽으로 결합한다.
문자 형에 숫자 형을 결합하면 문자 형으로 취급한다.

연산자(5/8)

- 비교 연산자 : 논리 문장에서 값들을 비교하는 용도로 사용

연산자	설명	예
==	값이 같으면 true	x == 1
===	값이 같고 유형도 같으면 true	X === 1
!=	값이 다르면 true	x != 2
!==	값이 다르거나 유형이 다르면 true	x !== 2
>	크면 true	x > 2
<	작으면 true	x < 2
>=	크거나 같으면 true	x >= 2
<=	작거나 같으면 true	x <= 2

연산자(6/8)

- 비교 연산자는 다음과 같이 조건문에서 많이 사용된다.

```
if (age > 18) {  
    msg = "입장하실 수 있습니다.";  
}
```

- 다음의 결과를 확인해보자.

```
<script>  
    var x = 10;  
    var y = 20;  
    document.write((x > y) + "<br>");  
    document.write((x < y) + "<br>");  
    document.write((x == y) + "<br>");  
    document.write((x != y) + "<br>");  
</script>
```

연산자(7/8)

- 논리 연산자
 - 여러 개의 조건을 조합하여 참인지 거짓인지를 따질 때 사용
 - 예를 들어 "비가 오지 않고 휴일이면 테니스를 친다."라는 문장에는 "비가 오지 않는다 " 라는 조건과 "휴일이다 " 라는 조건이 동시에 만족이 되면 테니스를 친다는 의미가 포함되어 있다.

연산자	사용 예	의미
&&	x && y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
	x y	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓
!	!x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참

연산자(8/8)

- 조건 연산자 (삼항 연산자)

```
maxValue = (x > y) ? x : y;
```

- $x > y$ 가 참이면 x 가 수식의 값이 된다.
- $x > y$ 가 거짓이면 y 가 수식의 값이 된다.

연산자 우선순위

우선순위	연산자	우선순위	연산자
1	. [] new	10	&
2	()	11	^
3	++ --	12	
4	! ~ + -(부호) typeof void delete	13	&&
5	* / %	14	
6	+ -(사칙연산자)	15	?: (삼항연산자)
7	<< >> >>>	16	yield
8	< <= > >= in instanceof	17	= += -= *= /= %=
9	== != === !==	18	<<= >>= >>>= &= ^= =
			,

prompt()

- 사용자에게 입력을 요청하는 대화 상자 소환 함수
 - [확인] 버튼 클릭 시 사용자가 입력한 값을 반환 받는다.
 - [취소] 버튼 클릭 시 null 값을 반환 받는다.

```
<script>  
    var result = prompt("대화상자에 표시할 텍스트 (required)",  
                        "기본 입력 텍스트 (optional)");  
</script>
```

필수 텍스트

옵션 텍스트

확인 취소

prompt() 덧셈 예제

```
<script>
  var firstVal, secondVal, input;

  input = prompt("첫번째 입력 값", "정수로 입력하세요");
  firstVal = parseInt(input);
  input = prompt("두번째 입력 값", "정수로 입력하세요");
  secondVal = parseInt(input);
  document.write(firstVal + secondVal + "<br>");
</script>
```

- Prompt() 를 이용한 덧셈 예제
 - 반환 받는 값의 타입은 string
 - 덧셈 연산을 진행할 때 string -> number 타입으로 변환 후 연산

getElementById()

- HTML 요소 접근 함수
 - 지정된 id 속성을 가진 요소를 반환한다.
 - 존재하지 않는 요소일 경우 null 값을 반환한다.

```
<script>  
    document.getElementById(elementId);  
</script>
```

- 다른 요소 접근 함수
 - getElementsByTagName()
 - getElementsByClassName()
 - querySelector()
 - querySelectorAll()

getElementById() 덧셈 예제

```
<script>
```

```
function calc(){  
    var fstVal = document.getElementById("firstVal").value;  
    var secVal = document.getElementById("secondVal").value;  
    var result;  
    result = parseInt(fstVal) + parseInt(secVal);  
    document.getElementById("sum").value = result;  
}
```

```
</script>
```

```
<body>
```

```
    첫번째 입력 값 : <input id="firstVal"> <br>
```

```
    두번째 입력 값 : <input id="secondVal"> <br>
```

```
    합계 : <input id="sum">
```

```
    <input type="button" value="계산" onclick="calc();">
```

```
</body>
```

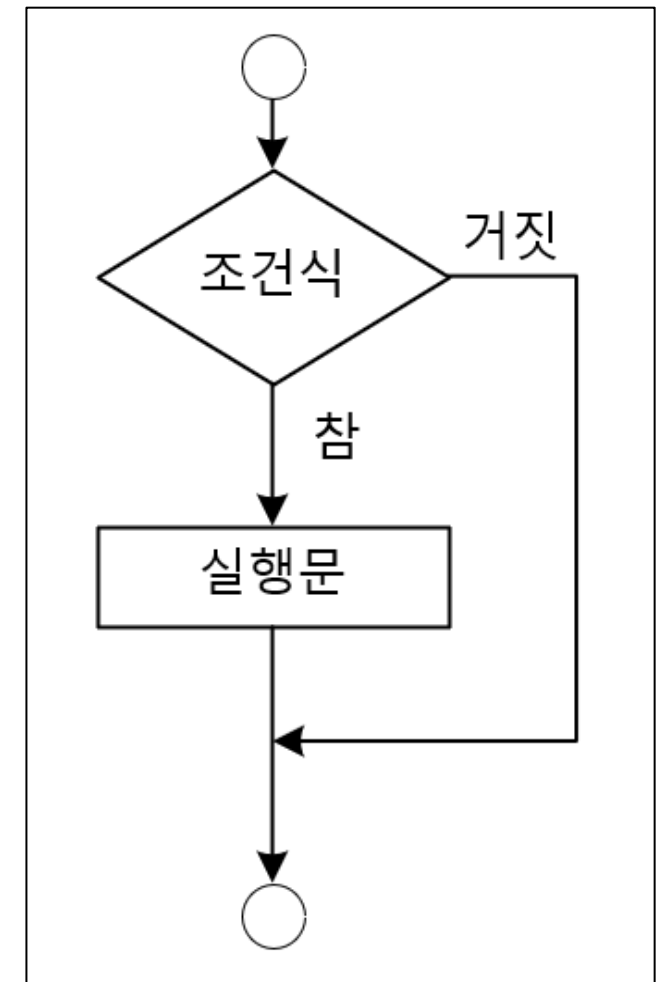
조건문(1/4)

- if/else : 조건식 결과에 따라 실행할 문장을 결정한다.
 - 조건식이 true인 경우 if 블록의 코드를 실행한다.
 - 동일한 조건식이 false인 경우 else 블록의 코드를 실행한다.

```
if (조건식) {  
    문장 1;  
}
```

조건이 참 일 때만 문장1이 실행된다.

형식	<pre>if (조건식) { 문장 1; } else { 문장 2; }</pre>
설명	만약 조건식이 참이면 문장1이 실행되고 그렇지 않으면 문장2가 실행된다.



조건문(2/4)

- else if : 연속적인 조건을 지정한다.
 - 첫번째 조건식이 false인 경우 새로운 조건을 지정한다.

```
<script>
  const time = new Date().getHours();
  let greeting;

  if (time < 10) {           // 10시 이전이면
    greeting = "Good Morning";
  } else if (time < 20) {    // 오후 8시 이전이면
    greeting = "Good day";
  } else {                  // 그렇지 않으면(오후 9시 이후이면)
    greeting = "Good evening";
  }

  alert(greeting);
</script>
```

If문 문제

- 숫자 2개와 연산자 1개를 입력 받아 연산자에 맞는 계산결과를 출력하는 프로그램을 작성하시오.

The image shows four overlapping 'Explorer 사용자 프롬프트' (Explorer User Prompt) windows, numbered 1 through 4, illustrating the steps of a program:

- 1** Explorer 사용자 프롬프트: 스크립트 프롬프트: 첫번째 숫자를 입력하세요. Input: 25. Button: 확인.
- 2** Explorer 사용자 프롬프트: 스크립트 프롬프트: 연산자를 입력하세요 (+, -, *, / 중 하나). Input: +. Button: 확인.
- 3** Explorer 사용자 프롬프트: 스크립트 프롬프트: 두번째 숫자를 입력하세요. Input: 63. Buttons: 확인, 취소.
- 4 결과** Explorer 사용자 프롬프트: 스크립트 프롬프트: 25+63 = 88. Buttons: ←, →.

조건문(3/4)

- switch : 조건에 따라 프로그램을 분기시키기 위해 사용된다.
- 제어식 값에 따라 실행할 문장을 결정하게 되므로 연속적인 if문보다 switch문을 사용하는 것이 좋다.

형식

```
switch(제어식) {  
    case c1:  
        문장1;  
        break;  
    case c2;  
        문장2;  
        break;  
    default:  
        문장d;  
        break;  
}
```


조건문(4/4)

```
<script>  
  let text;  
  
  switch (new Date().getDay()) {  
    case 6:  
      text = "Today is Saturday";  
      break;  
    case 0:  
      text = "Today is Sunday";  
      break;  
    default:  
      text = "Looking forward to the Weekend";  
  }  
  
  alert(text);  
</script>
```

switch문 문제

- 점수를 입력받아 학점을 출력하시오. (switch문을 이용)
 - 점수가 90 ~ 100이면 'A'
 - 점수가 80 ~ 89이면 'B'
 - 점수가 70 ~ 79이면 'C'
 - 점수가 60 ~ 69이면 'D'
 - 점수가 0 ~ 59이면 'F'
 - 출력은 document.write()를 이용

조건문 문제

- 두 사람의 가위, 바위, 보를 입력 받아 승자를 출력하는 프로그램을 작성하시오.

Explorer 사용자 프롬프트

스크립트 프롬프트:
첫번째 사람의 가위 바위 보를 입력하세요

가위

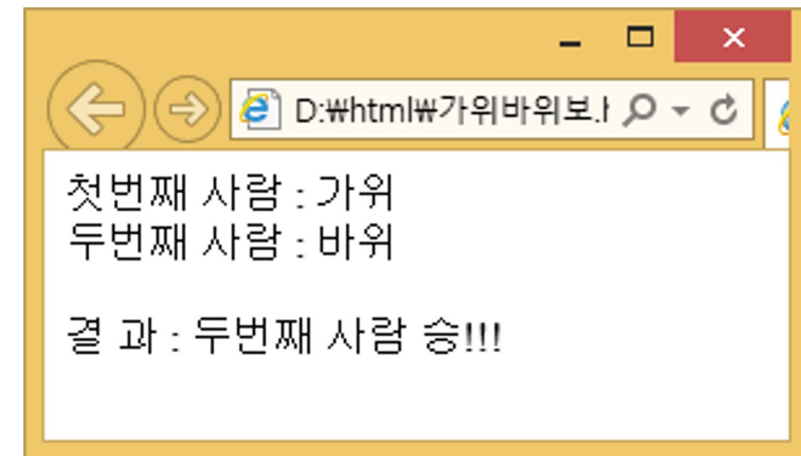
확인 취소

Explorer 사용자 프롬프트

스크립트 프롬프트:
두번째 사람의 가위 바위 보를 입력하세요

바위

확인 취소



반복문(1/5)

- while : 지정된 조건이 참일 때 반복 실행한다.

```
<body>
```

```
  <p id="result"></p>
```

```
</body>
```

```
<script>
```

```
  let text = "";
```

```
  let i = 0;
```

```
  while (i < 10) {
```

```
    text += "<br>The number is" + i;
```

```
    i++;
```

```
  }
```

```
  document.getElementById("result").innerHTML = text;
```

```
</script>
```

JavaScript While Loop

The number is 0

The number is 1

The number is 2

The number is 3

The number is 4

The number is 5

The number is 6

The number is 7

The number is 8

The number is 9

반복문(2/5)

- do while : 조건을 확인하기 전 1회 실행 후 지정된 조건이 참일 때 반복 실행한다.

```
<body>
  <p id="result"></p>
</body>

<script>
  let text = "";
  let i = 0;
  do {
    text += i + "<br>";
    i++;
  }
  while (i < 5);
  document.getElementById("result").innerHTML = text;
</script>
```

JavaScript Statements

The do..while Loop

0
1
2
3
4

반복문(3/5)

- for : 조건식이 참인 동안 코드 블록을 반복 실행한다.

```
<body>  
  <p id="result"></p>  
</body>
```

```
<script>  
  let text = "";  
  for (let i = 0; i < 5; i++) {  
    text += "The number is " + i + "<br>";  
  }  
  document.getElementById("result").innerHTML = text;  
</script>
```

JavaScript For Loop

```
The number is 0  
The number is 1  
The number is 2  
The number is 3  
The number is 4
```

반복문(4/5)

- nested loop (중첩 반복문)

```
<style>
```

```
table, td {border:1px solid black;}
```

```
</style>
```

```
<script>
```

```
document.write("<h1>구구단표</h1>");
```

```
document.write("<table>");
```

```
for (var i = 1; i <= 9; i++) {
```

```
    document.write("<tr>");
```

```
    document.write("<td>" + i + "</td>");
```

```
    for (var j = 2; j <= 9; j++) {
```

```
        document.write("<td>" + i * j + "</td>");
```

```
    }
```

```
    document.write("</tr>");
```

```
}
```

```
document.write("</table>");
```

```
</script>
```

구구단표

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

반복문(5/5)

- for / in : 해당 객체의 모든 열거 가능한 속성을 순회한다.
 - 객체의 열거할 수 있는 속성 이름을 지정 변수에 대입
 - 대입 받은 변수를 이용해 반복문 안에서 속성에 순차적으로 접근

```
<script>  
  let myCar = { make: "BMW", model: "X5", year: 2013 };  
  let txt = "";  
  for (let x in myCar) {  
    txt += myCar[x] + " ";  
  }  
  document.write(txt);  
</script>
```

JavaScript For In Loop

The for in statement loops through the properties of an object:

BMW X5 2013

break 문장

- 반복문을 벗어나기 위해 사용한다.
- 반복문 안에서 break문장 실행 시 해당 영역을 빠져나온다.

```
<script>
  var msg = "";
  for (var i = 0; i < 10; i++) {
    if (i == 3) {
      break;
    }
    msg += i + "<br>";
  }
  document.write(msg);
</script>
```

A loop with a **break** statement.

0
1
2

continue 문장

- 현재 실행하고 있는 반복 과정 중 지정된 조건에 대해서 생략하고 다음 반복문을 이어 진행한다.

```
<script>
  var msg = "";
  for (var i = 0; i < 10; i++) {
    if (i == 3) {
      continue;
    }
    msg += i + "<br>";
  }
  document.write(msg);
</script>
```

A loop with a **continue** statement.

A loop which will skip the step where $i = 3$.

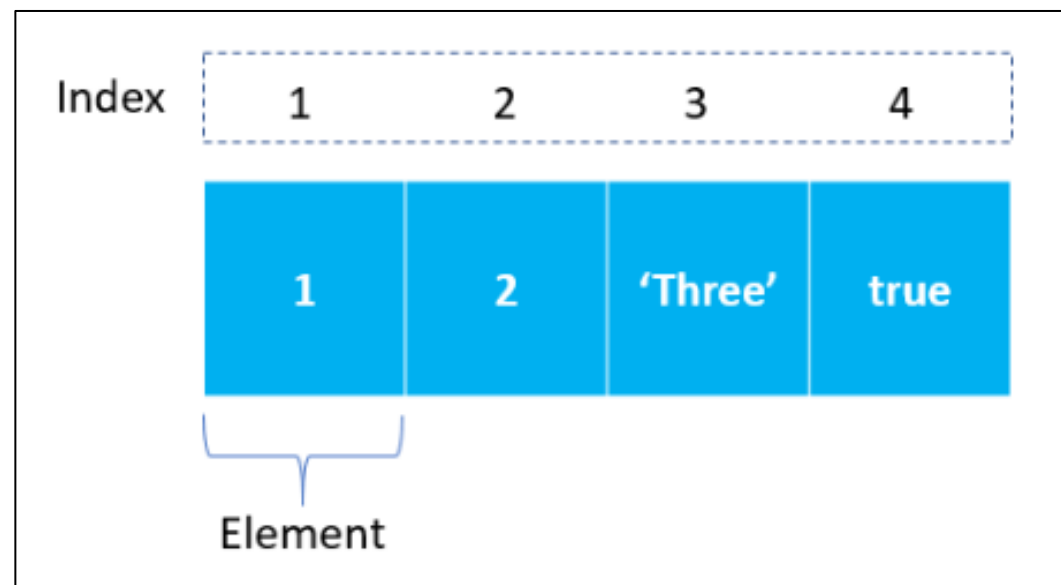
0
1
2
4
5
6
7
8
9

문제

1. 1부터 10까지의 합을 구하는 프로그램을 작성하시오.
2. 1부터 200까지의 짝수의 합을 구하는 프로그램을 작성하시오.(continue를 이용)
3. 사용자가 입력한 값을 계속 더하고, 사용자가 0을 입력하면 그때까지 누적된 값을 출력하는 프로그램을 작성하시오.
4. 다중 for문을 이용해서 1~ 10 까지 중
i와 k의 더한 합이 3의 배수일 때만 출력 continue를 이용
5. 1~100 까지 중 2의 배수이면서 3의 배수인 것만 출력
6. 두 수를 입력(prompt) 두수의 합이 100이상일때만 출력
(continue를 이용 , 두수 모두 0 이 입력되면 종료)

배열(1/3)

- 배열은 하나 이상의 값을 가질 수 있는 특수 변수이다.
- 배열을 구성하는 각각의 값을 배열 요소(element)라고 한다.
- 배열의 위치를 가리키는 숫자를 인덱스(index)라고 하며, 인덱스를 참조하여 값에 접근할 수 있다.
- 같은 배열 안의 요소 타입이 서로 다를 수 있는 특징이 있다.



배열(2/3)

- 배열 생성 방법

- 1) 리터럴로 배열 생성

- `const fruits = ["apple", "banana", "peach"];`

- 2) Array 객체로 배열 생성

- `const fruits = [];`
 `fruits[0] = "Apple";`
 `fruits[1] = "Banana";`
 `fruits[2] = "Orange";`- `const fruits = new Array("apple", "banana", "orange");`

*가독성 및 실행 속도 면에서 더 나은 리터럴 방식을 권장

배열 (3/3)

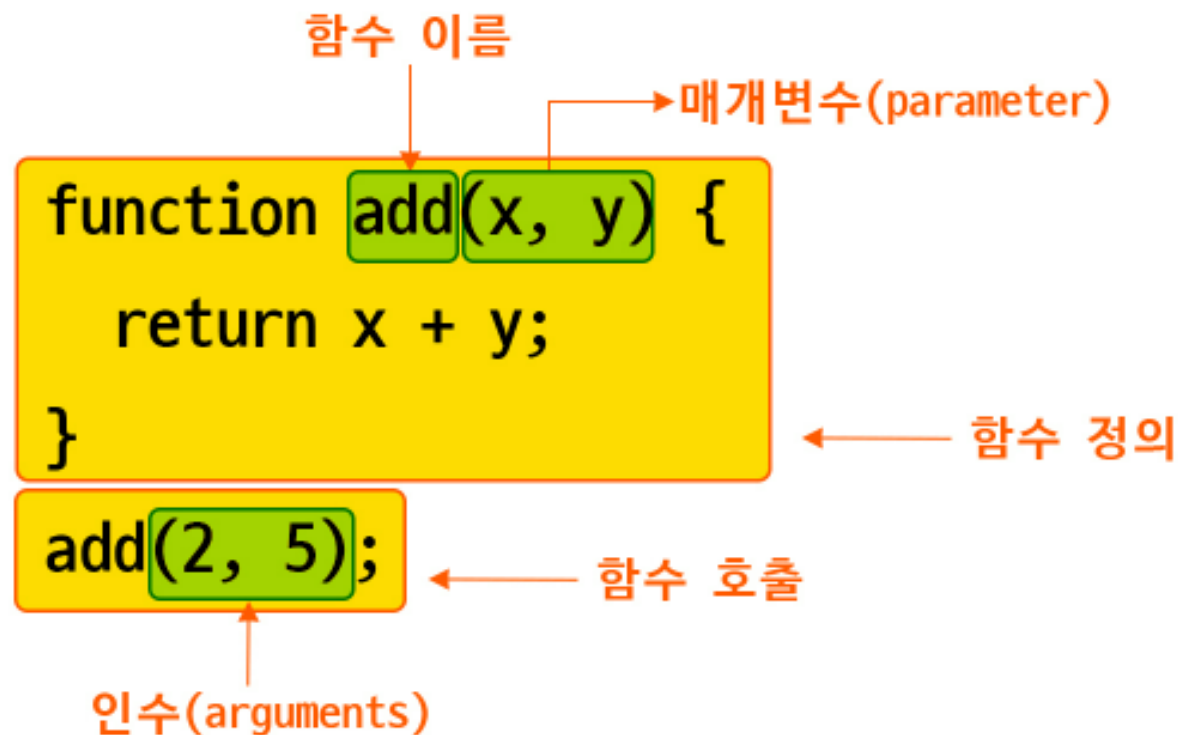
```
<script>
  const fruits = [];
  fruits[0] = "Apple";
  fruits[1] = "Banana";
  fruits[2] = "Orange";

  for (i = 0; i < fruits.length; i++) {
    document.write(fruits[i] + "<br>");
  }

  for(x in fruits ){
    document.write(fruits[x] + "<br>");
  }
</script>
```

함수

- 함수란 호출을 통해 실행되어 특정한 작업을 수행하고 그 결과를 반환하는 코드 블록이다.
- 함수 명을 지정할 때 변수와 동일한 규칙을 사용한다.
- 괄호에 심표로 구분된 매개변수 이름이 포함될 수 있으며, 이는 함수 호출 시 인수를 통해 전달되는 값을 받는다.



함수 만들기

1. 파라미터도 있고 반환 값도 있는 함수

```
function 함수 명(파라미터1, 파라미터2, ...) {  
    return 반환 값;  
}
```

2. 파라미터는 있고 반환 값은 없는 함수

```
function 함수 명(파라미터1, 파라미터2, ...) {  
    명령문;  
}
```

3. 파라미터는 없고 반환 값은 있는 함수

```
function 함수 명() {  
    return 반환 값;  
}
```

4. 파라미터도 없고 반환 값도 없는 함수

```
function 함수명() {  
    명령문;  
}
```


함수의 호출과 인수, 매개변수

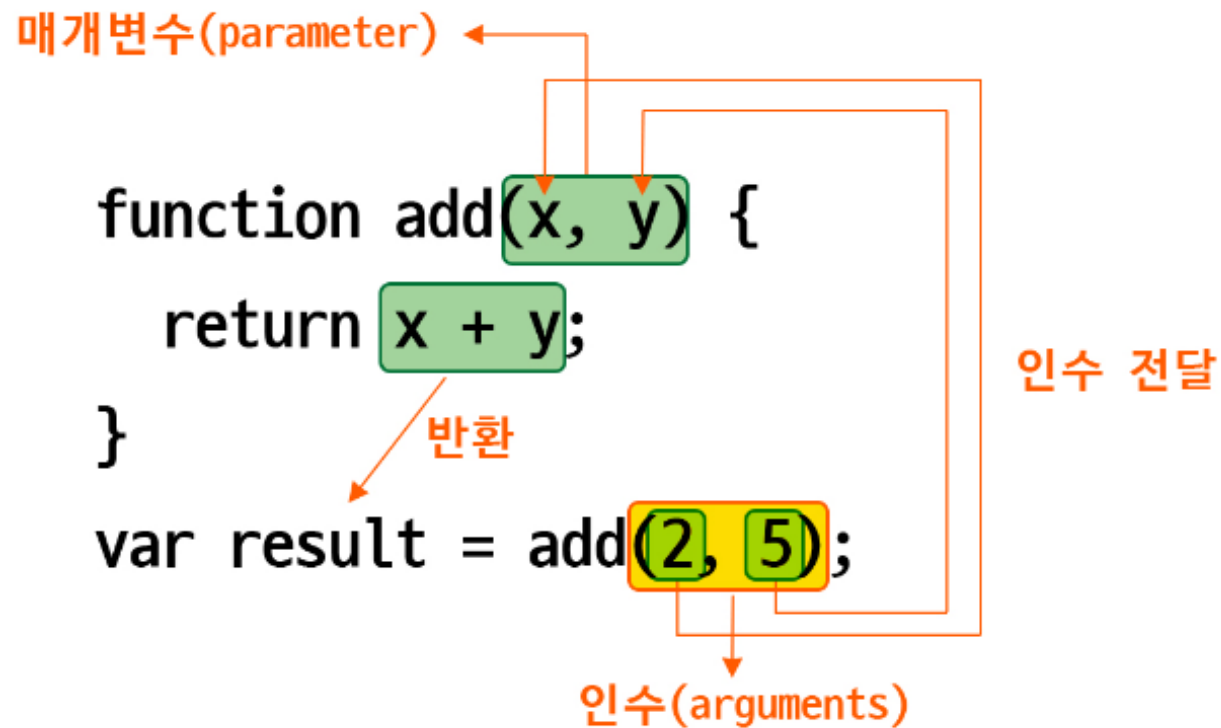
- 함수는 호출에 의해서 실행

```
function showDialog(para1, para2) { 매개변수, 가인수
  명령문1;
  명령문2;
}
showDialog(arg1, arg2); 인수, 실인수
```

- 인수(argument) : 함수에 전달할 값을 담은 변수 또는 상수
- 인수는 데이터 타입이 없으며 개수에도 제약이 없다.
- 매개변수(parameter) : 함수 호출 시 인수로 전달된 값을 함수 내부에서 사용할 수 있도록 선언된 변수
- 실 인수는 남으면 무시되고, 모자라는 가 인수는 undefined 된다.

함수의 반환 값

- return 문장을 사용하여 실행 결과를 외부로 반환할 수 있다.



- 반환 값을 변수에 저장하지 않고 바로 수식에 사용해도 된다.
 - `window.onload = function(){
 document.getElementById("result").innerHTML = add(2,5);
}`

함수 예제(1/2)

```
<body>
  <form>
    첫째 :<input id="x"><br>
    둘째 :<input id="y"><br>
    결과 :<input id="sum"><br>
    <input type="button" onclick="calc(+)" value="+">
    <input type="button" onclick="calc(-)" value="-">
    <input type="button" onclick="calc(*)" value="*">
    <input type="button" onclick="calc(/)" value="/"><br>
    <p>첫째 값 :<span id="sp1"></span> </p>
    <p>둘째 값 :<span id="sp2"></span> </p>
    <p>결과 :<span id="sp3"></span> </p>
  </form>
</body>
```

함수 예제(2/2)

```
<script>
```

```
function calc(a){
```

1. 입력 받은 값을 가져온다. (value 속성 사용)
2. a의 값을 비교하여 각 계산을 수행하는 함수를 호출한다.
3. 리턴 받은 결과 값을 출력한다. (innerHTML 사용)

```
}
```

```
function f_add() {
```

```
  return ...
```

```
}
```

```
function f_sub() {
```

```
  return ...
```

```
}
```

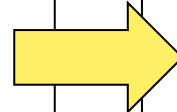
```
...
```

```
</script>
```

무명 함수

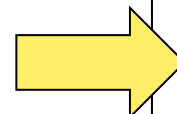
- 무명 함수 또는 익명 함수(anonymous function)
 - 표현식 함수를 변수에 저장하는 형태로 변수 이름으로 호출한다.

```
// 선언식 함수(기명 함수)  
function showDialog(str) {  
    alert(str);  
}  
showDialog("안녕하세요.");
```



```
// 표현식 함수(익명 함수)  
let showDialog = function (str) {  
    alert(str);  
}  
showDialog("안녕하세요.");
```

- 함수를 한번만 바로 사용할 때 익명의 자체 호출 함수를 사용한다.



```
// 익명 (자체 호출) 함수  
(function (str) {  
    alert(str);  
})("안녕하세요.");
```

alert()

- 메시지와 [확인] 버튼이 있는 경고 상자를 띄우는 함수
 - 사용자에게 정보를 전달할 때 사용된다.

```
<button onclick="myAlert()">Try it</button>
```

```
</script>
```

```
function myAlert(){  
    alert("Hello! I am an alert box!");  
}  
</script>
```

Hello! I am an alert box!

확인

confirm()

- 메시지와 [확인],[취소] 버튼이 있는 대화 상자를 띄우는 함수
 - [확인] 버튼 클릭 시 true를 반환한다.
 - [취소] 버튼 클릭 시 false를 반환한다.

```
<body>
  <button onclick="myFunction()">Try it</button>
  <p id="demo"></p>
<script>
function myFunction() {
  let text = "Press a button!\nEither OK or Cancel.";
  if (confirm(text) == true) {
    text = "You pressed OK!";
  } else {
    text = "You canceled!";
  }
  document.getElementById("demo").innerHTML += text;
}
```

You pressed OK!

You canceled!

Press a button!
Either OK or Cancel.

확인

취소