

HTML5 + CSS3 + JavaScript

로 배우는 웹프로그래밍 기초

개정
2판



제1장 자바 스크립트 기초

HTML5 기술의 핵심

웹 페이지의
내용은
내가 책임집니다.



HTML

웹 페이지의
표현은
내가 책임집니다.



CSS

웹 페이지의
동작은
내가 책임집니다.



JAVASCRIPT

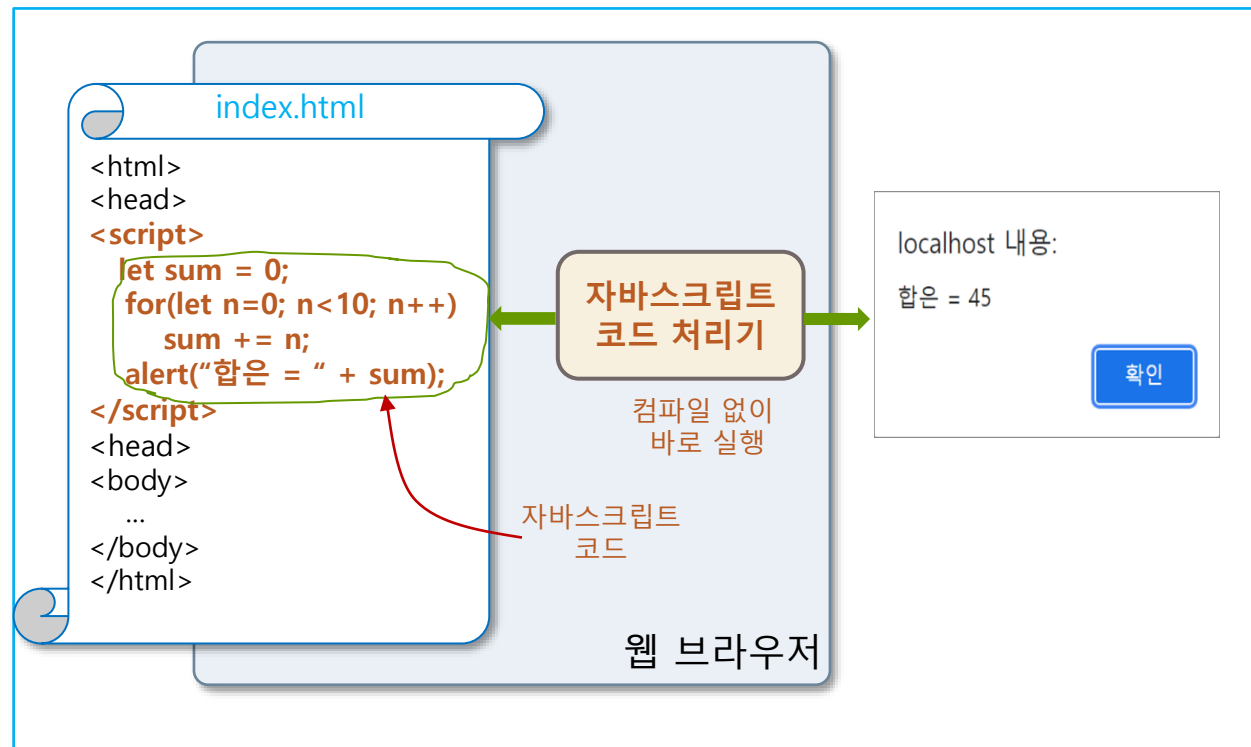
자바 스크립트 소개

- Javascript

- 1995년 넷스케이프 개발
- Netscape Navigator 2.0 브라우저에 최초 탑재
- 웹 프로그래밍 개념 창시

- 특징

- HTML 문서에 내장
 - 조각 소스 코드
- 스크립트 언어
 - 인터프리터 실행
 - 컴파일 필요 없음
- 단순
 - C언어 구조 차용
 - 배우기 쉬움



자바 vs 자바 스크립트

특징	자바 언어	자바스크립트
언어 종류	소스 파일을 컴파일하여 실행하는 컴파일 언어이다.	브라우저가 소스 코드를 직접 해석하여 순차적으로 실행하는 인터프리트 언어이다.
실행 방식	자바 가상 기계 위에서 실행한다.	브라우저 위에서 실행된다.
작성 위치	별도의 소스 파일에 작성	HTML 파일 안에 삽입 가능
변수 선언	변수의 타입을 반드시 선언하여야 함	변수의 타입을 선언하지 않아도 사용가능
유사점	C언어와 비슷한 구문, 둘다 객체 지향을 지원한다.	

웹 페이지에서 자바스크립트의 역할

- 사용자의 입력 및 계산
 - 마우스와 키보드 입력은 오직 자바스크립트로만 가능
 - 계산 기능
- 웹 페이지 내용 및 모양의 동적 제어
 - HTML 태그의 속성, 콘텐츠, CSS 프로퍼티 값 동적 변경
- 브라우저 제어
 - 브라우저 윈도우 크기와 모양 제어
 - 새 윈도우 열기/닫기
 - 다른 웹 사이트 접속
 - 히스토리 제어
- 웹 서버와의 통신
- 웹 애플리케이션 작성
 - 캔버스 그래픽, 로컬/세션 스토리지 저장, 위치정보서비스 등

자바스크립트 코드의 위치

- 자바스크립트 코드 작성이 가능한 위치

1. HTML 태그의 **이벤트 리스너** 속성에 작성

2. `<script>` `</script>` 태그에 작성

3. 자바스크립트 파일에 작성

4. URL 부분에 작성

이벤트가 발생할때까지
기다린후 이벤트에 응답하는
javascript의 함수



HTML태그의 이벤트 리스너 속성에 작성

onclick 이벤트
리스너 속성

자바스크립트 코드
(이미지를 banana.png로 교체)

```

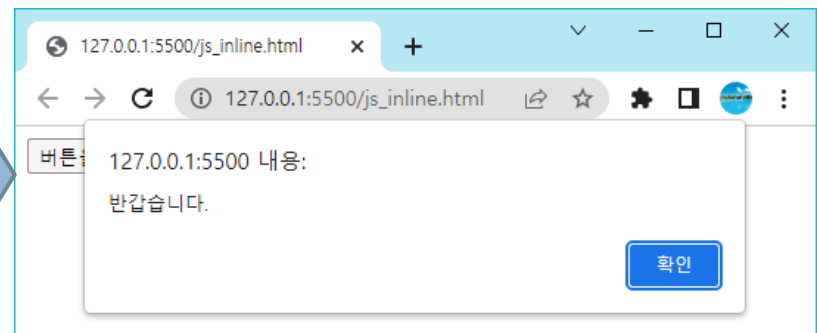
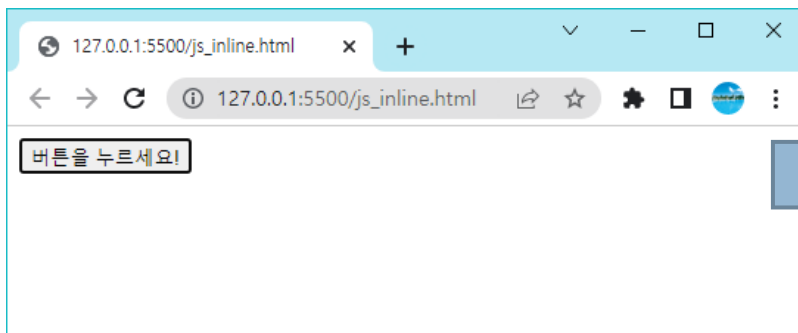
```

인라인 자바스크립트

js_inline1.html

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

onclick 이벤트가 발생하면
alert()를 호출한다.



Js_inline2.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<script>
```

```
function paragraph_over() {
```

```
    paragraph.innerText = "안녕하세요^^ 반가워요.";
```

```
    paragraph.style.color="Red"; }
```

```
function paragraph_out() {
```

```
    paragraph.innerText = "마우스를 여기에 위치해보세요.";
```

```
    paragraph.style.color="Black"; }
```

```
</script>
```

innerText는 태그와 태그 사이에 존재하는 텍스트 데이터를 조작할 수 있도록 하는 속성

```
<body>
```

```
<p id=paragraph onmouseover=paragraph_over()
```

```
    onmouseout=paragraph_out() >
```

```
    마우스를 여기에 위치해보세요.
```

```
</p>
```

```
</body>
```

```
</html>
```

Js_inline3.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p>마우스를 이미지 위로 올리면 이미지가 커집니다.</p>
```

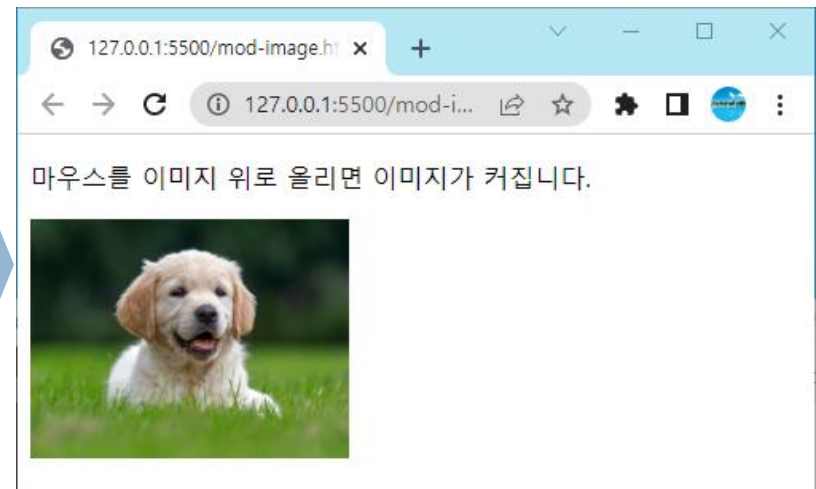
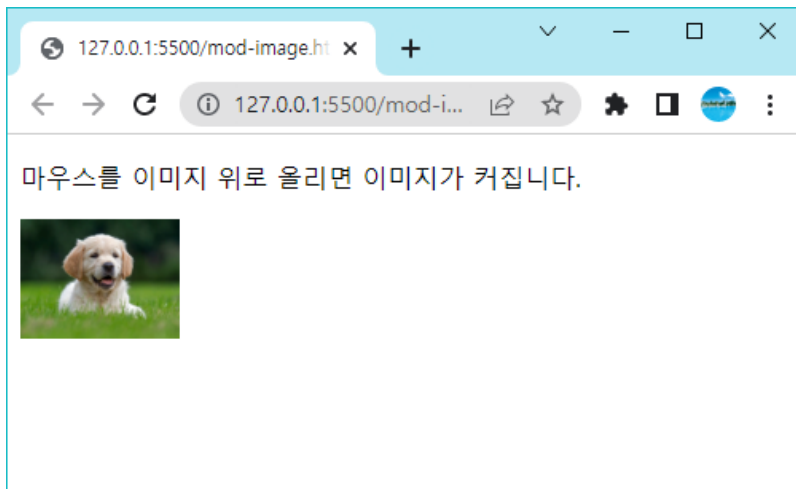
```

```

```
</body>
```

```
</html>
```

마우스의 커서가 해당 요소위에
위치할때 또는 빠져나갈때 일어나는
이벤트



Js_inline4.html

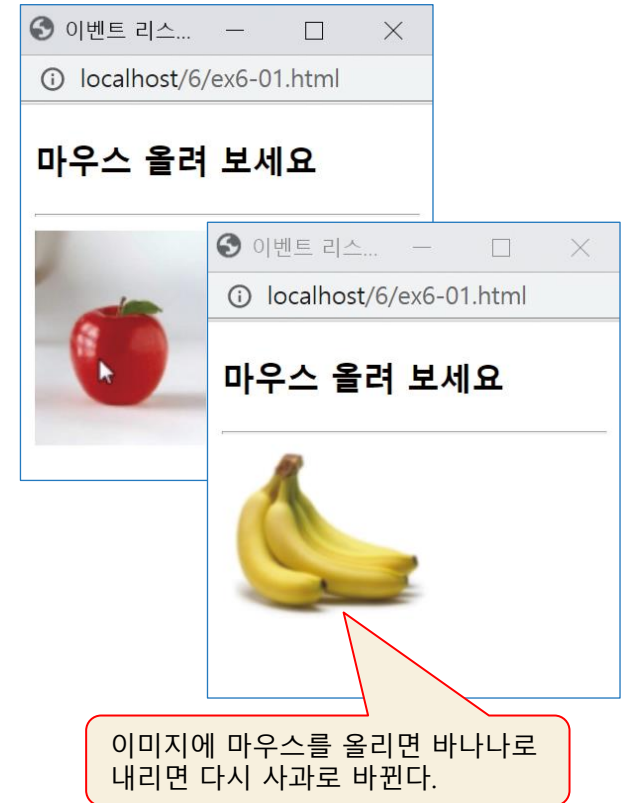
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>이벤트 리스너 속성에 자바스크립트
코드</title>
</head>
<body>
<h3>마우스 올려 보세요</h3>
<hr>

</body>
</html>
```

이벤트 리스너
속성

this는 현재 img 태그를
가리키는 자바스크립트 키워드

자바스크립트
코드

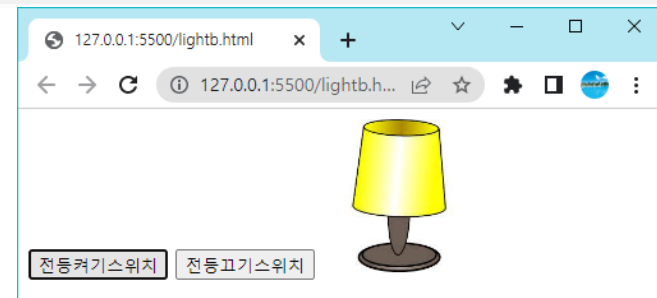
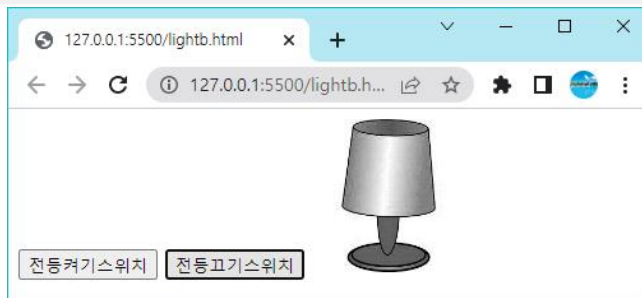


Js_inline5.html

```
<!DOCTYPE html>
<html>
<body>
<button
onclick= "document.getElementById('lightb').src='media/light-bulb1.png'">
    전등켜기스위치 </button>
<button
onclick= "document.getElementById('lightb').src='media/light-bulb2.png'">
    전등끄기스위치 </button>

</body>
</html>
```

이것이 자바스크립트이다.



<script> </script> 태그에 자바스크립트 작성

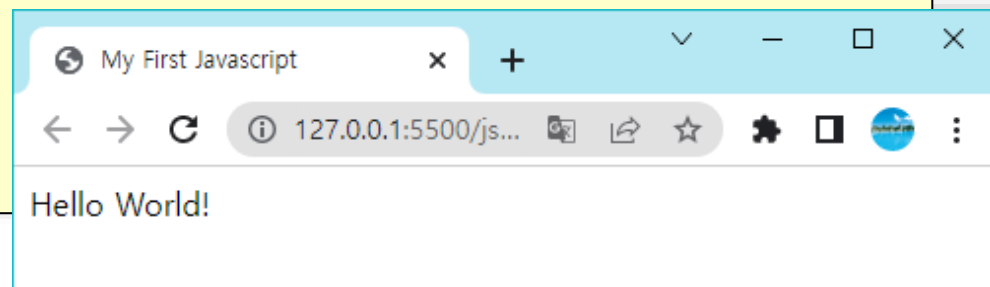
- 특징
 - <head> </head>나 <body> </body> 내 어디든 가능
 - 웹 페이지 내에 여러 번 삽입 가능

내부 자바 스크립트

js_script1.html

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

document.write() 는 document 객체의 write() 메소드를 호출하는 문장이다. 객체는 속성과 동작을 한데 모아둔 것으로 메소드는 동작에 해당된다. document 객체는 웹페이지를 나타내는 객체이다. document.write()는 동적으로 HTML 파일의 콘텐츠를 생성하는 메소드이다.



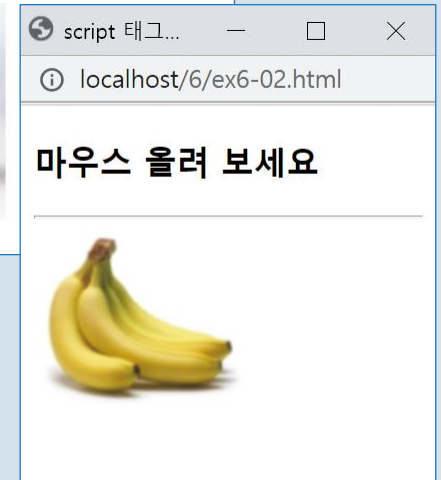
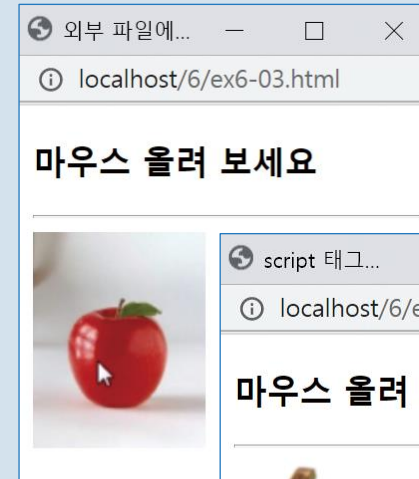
js_script2.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>script 태그에 자바스크립트 작성</title>
<script>
function over(obj) {
  obj.src="media/banana.png";
}
function out(obj) {
  obj.src="media/apple.png";
}
</script>
</head>
<body>
<h3>마우스 올려 보세요</h3>
<hr>

</body>
</html>
```

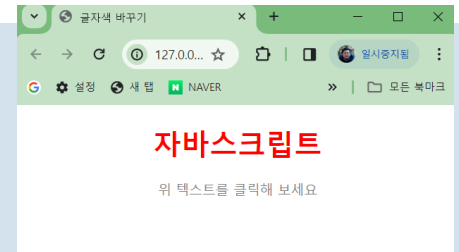
obj는 전달받은
img 태그를 가리킴

this는 현재 img 태그를
가리키는 자바스크립트키워드



js_script3html

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>글자색 바꾸기</title>
  <style>
    body { text-align:center; }
    #heading { color:blue; }
    #text { color:gray; font-size:15px; }
  </style>
</head>
<body>
  <h1 id="heading">자바스크립트</h1>
  <p id="text">위 텍스트를 클릭해 보세요</p>
  <script>
    var heading = document.querySelector('#heading');
    heading.onclick = function() {
      heading.style.color = "red"; }
  </script>
</body>
</html>
```



querySelector() 함수

- 괄호속에 제공한 선택자와 일치하는 문서내 첫번째 Element를 반환
- id, class, 복합태그 모두 가능
- getElementById() 함수와 비슷

자바스크립트 코드를 별도 파일에 작성

- 자바스크립트 코드 파일 저장
 - 확장자 .js 파일에 저장
 - <script> 태그 없이 자바스크립트 코드만 저장
- 여러 웹 페이지에서 불러 사용
 - 웹 페이지마다 자바스크립트 코드 작성 중복 불필요
 - <script> 태그의 src 속성으로 파일을 불러 사용

```
<script src="파일이름.js">
```

```
// HTML5부터 이곳에 자바스크립트 코드 추가 작성하면 안 됨
```

```
</script>
```

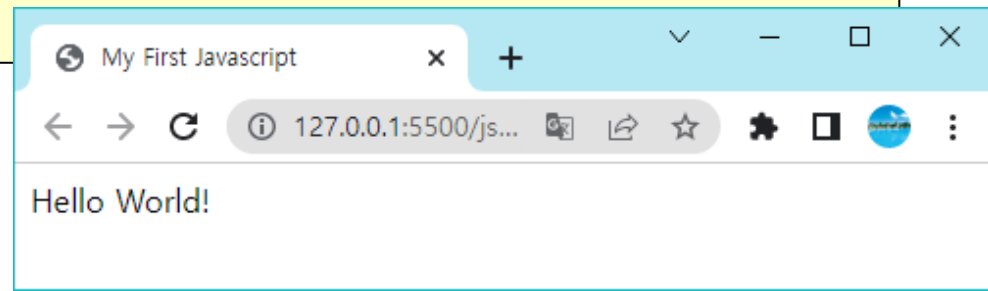
외부 자바스크립트

js_external1.html

```
<!DOCTYPE html>
<html>
<head>
  <script src="myscript.js"> </script>
</head>
<body>
</body>
</html>
```

myscript.js

```
document.write("Hello World!");
```



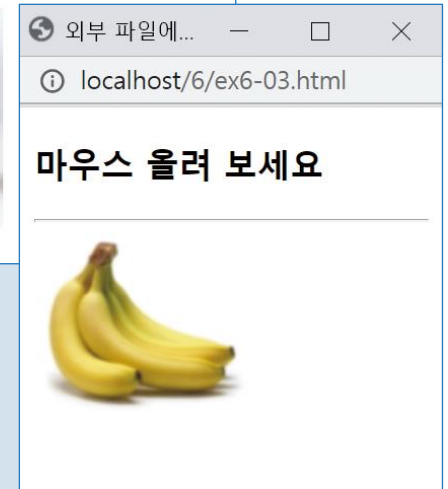
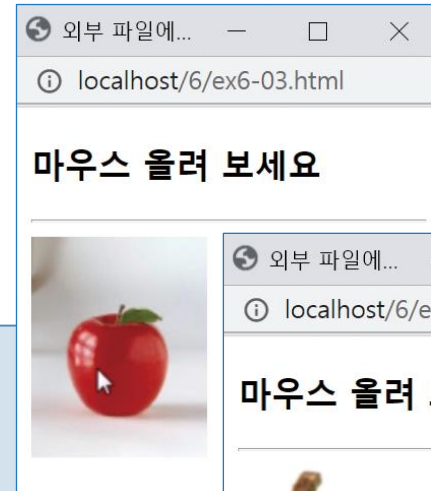
```
/* 자바스크립트 파일 lib.js */  
function over(obj) {  
    obj.src="media/banana.png";  
}  
function out(obj) {  
    obj.src="media/apple.png";  
}
```

lib.js

lib.js
불러오기

js_external2.html

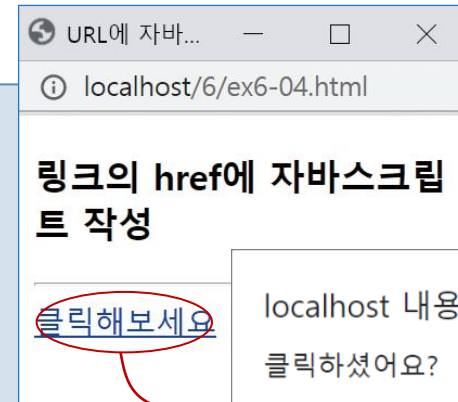
```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="utf-8">  
<title>외부 파일에 자바스크립트 작성</title>  
<script src="lib.js">  
</script>  
</head>  
<body>  
<h3>마우스 올려 보세요</h3>  
<hr>  
  
</body>  
</html>
```



링크의 href에 자바스크립트 코드 작성

js_href.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>URL에 자바스크립트 작성</title>
</head>
<body>
  <h3>링크의 href에 자바스크립트 작성</h3>
  <hr>
  <a href="javascript:alert('클릭하셨어요?')">
    클릭해보세요</a>
</body>
</html>
```



localhost 내용:
클릭하셨어요?

확인

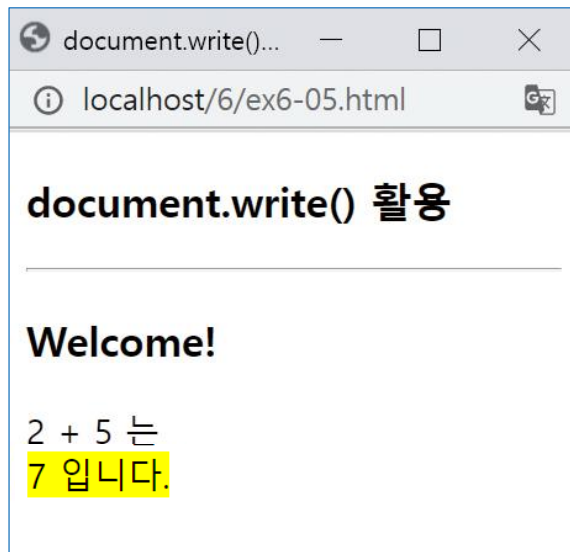
자바스크립트로 HTML 콘텐츠 출력

- 자바스크립트로 HTML 콘텐츠를 웹 페이지에 직접 삽입
 - 바로 브라우저 윈도우에 출력
 - document.write()
예) document.write("<h3>Welcome!</h3>");
 - document.writeln()
 - writeln()은 텍스트에 'wn' 을 덧붙여 출력
 - 'wn'을 덧붙이는 것은 고작해야 빈칸 하나 출력
 - 다음 줄로 넘어가는 것은 아님

js_write1.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>document.write() 활용 </title>
</head>
<body>
<h3>document.write() 활용 </h3>
<hr>
<script>
  document.write("<h3>Welcome!</h3>");
  document.write("2 + 5 는 <br>");
  document.write("<mark>7 입니다.</mark>");
</script>
</body>
</html>
```

문서에서 주의를 끌기 위한 강조



자바스크립트 다이얼로그 : **prompt()** 다이얼로그

- **prompt("메시지", "디폴트 입력값")** 함수
 - 사용자로부터 문자열을 입력 받아 리턴

```
<script>
```

```
let ret = prompt("이름을 입력하세요", "황기태");
```

```
if(ret == null) {
```

```
    // 취소 버튼이나 다이얼로그를 닫은 경우
```

```
}
```

```
else if(ret == "") {
```

```
    // 문자열 입력 없이 확인 버튼 누른 경우
```

```
}
```

```
else {
```

```
    // ret에는 사용자가 입력한 문자열
```

```
}
```

```
</script>
```

localhost 내용:

이름을 입력하세요

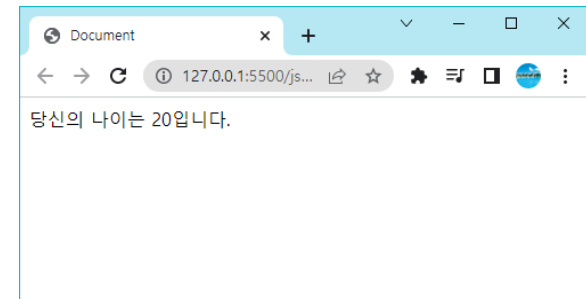
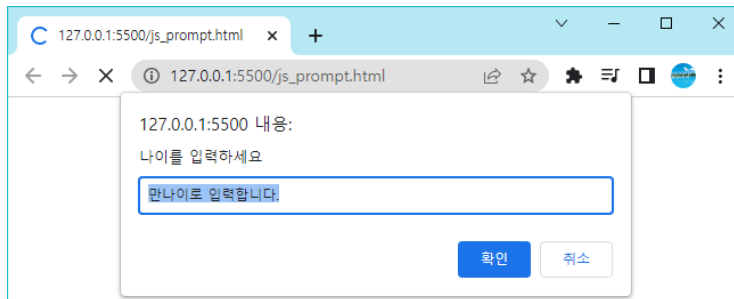
확인

취소

js_prompt.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>prompt() 활용</title>
</head>
<body>
  <script>
    let age = parseInt(prompt("나이를 입력하세요", "만나이로 입력합니다.));
    document.write("당신의 나이는 " + age + "입니다.");
  </script>
</body>
</html>
```

문자열을 숫자로 변환하기

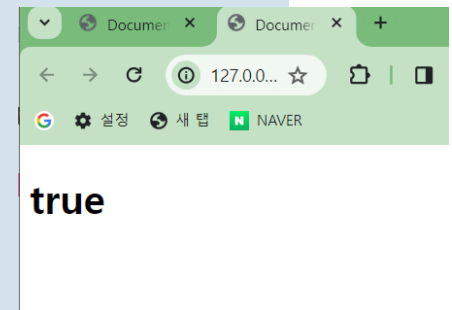
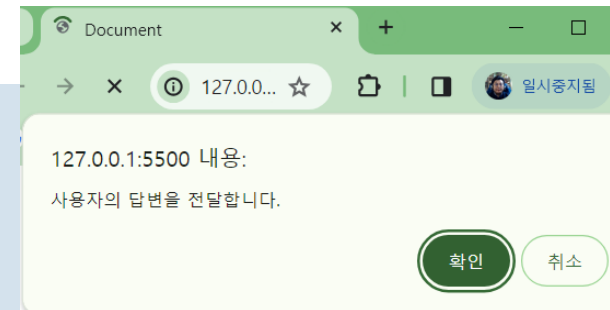


자바스크립트 다이얼로그 : 확인 다이얼로그

- **confirm("메시지") 함수**
 - "메시지"를 출력하고 '확인/취소(OK/CANCEL)'버튼을 가진 다이얼로그 출력
 - '확인' 버튼을 누르면 true, '취소' 버튼이나 강제로 다이얼로그를 닫으면 false 리턴

js_confirm.html

```
<script>
  let user = confirm("사용자의 답변을 전달합니다.");
  if(user == true) {
    document.write("<h1>" + user + "</h1>")
  }
  else {
    document.write("<h1>" + user + "</h1>")
  }
</script>
```



자바스크립트 다이얼로그 : 경고 다이얼로그

- **alert("메시지") 함수**

- 메시지'와 '확인' 버튼을 가진 다이얼로그 출력, 메시지 전달

alert("클릭하였습니다.");

js_alert.html

```
<!DOCTYPE html>
<html>
<body>
  <script>
    let name = prompt("이름을 입력해 주세요.", "");
    alert(name);
  </script>
</body>
</html>
```

localhost 내용:
클릭하였습니다.

확인

자바스크립트 식별자

- 식별자

- 자바스크립트 프로그램의 변수, 상수(리터럴), 함수의 이름
- 식별자 만드는 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

- 문장

- 자바스크립트 프로그램의 기본 단위는
- 문장과 문장을 구분하기 위해 세미콜론(;) 사용

```
i = i + 1           // (O) 한 줄에 한 문장만 있는 경우 세미콜론 생략 가능  
j = j + 1;          // (O)  
k = k + 1; m = m + 1; // (O) 한 줄에 여러 문장  
n = n + 1; p = p + 1; // (X) 첫 번째 문장 끝에 세미콜론이 필요함
```

- 주석문

```
// 한 라인 주석. 라인의 끝까지 주석 처리  
/*  
    여러 라인 주석  
*/
```

데이터 타입

- 자바스크립트 언어에서 다루는 데이터 종류
 - 숫자 타입 : 정수, 실수(예: 42, 3.14)
 - 논리 타입 : 참, 거짓(예: true, false)
 - 문자열 타입(예: '좋은 세상', "a", "365", "2+4")
 - 객체 레퍼런스 타입 : 객체를 가리킴. C 언어의 포인터와 유사
 - null : 값이 없음을 표시하는 특수 키워드. Null, NULL과는 다름
- 특징
 - 자바스크립트에는 문자 타입 없음. 문자열로 표현

변수

- 변수
 - 자바스크립트 프로그램이 실행 중에 데이터를 저장하는 공간
- 변수 선언
 - 변수 이름을 정하고, 저장 공간 할당
 - 3가지 방법(현재 3가지 방법 모두 사용)
 - **var 키워드 이용**
 - 자바스크립트 언어가 도입될 때부터 있었고 지금도 사용
 - **let 키워드 이용**
 - var의 문제점을 해소하기 위해 2015년 자바스크립트 언어 표준 (ECMAScript, ES6)에서 새로 추가
 - **const 키워드 이용**
 - 2015년 자바스크립트 언어 표준(ECMAScript, ES6)에서 새로 추가
 - 상수 선언(한 번 저장되면 값을 바꾸지 못함)

변수 선언 사례

- var로 선언

```
var score;           // 변수 score 선언
var year, month, day; // year, month, day의 3 개의 변수 선언
var address = "서울시"; // address 변수를 선언하고 "서울시"로 초기화
```

- let으로 선언

```
let score;           // 변수 score 선언
let year, month, day; // year, month, day의 3 개의 변수 선언
let address = "서울시"; // address 변수를 선언하고 "서울시"로 초기화
```

- var나 let 없이 선언

```
age = 21;           // var나 let 없이 변수 age가 선언. 동시에 21로 초기화
```

- 자바스크립트에는 변수 타입 없음

- 변수 타입 선언하지 않음

```
let score; // 정상적인 변수 선언
int score; // 오류. 변수 타입 int 없음
```

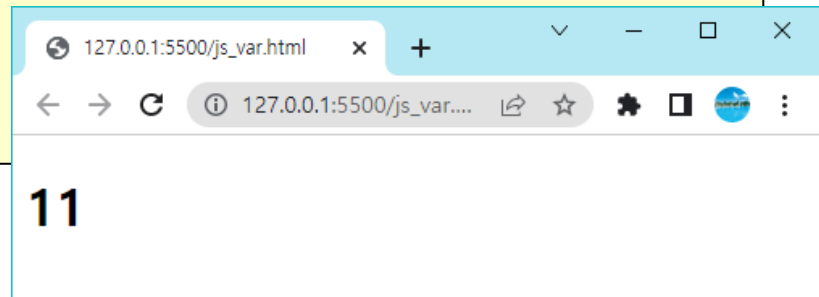
- 변수에 저장되는 값에 대한 제약 없음

```
score = 66.8; // 실수도 저장 가능
score = "high"; // 문자열로 저장 가능
```

js_var.html

```
<!DOCTYPE html>
<html>
<body>
  <h1 id="test"> </h1>
  <script>
    let x = 5;
    let y = 6;
    let z = x + y;
    document.getElementById("test").innerHTML = z;
  </script>
</body>
</html>
```

아이디 test를 가진 요소를
찾아서 요소 안의 콘텐츠를 변수
z의 값으로 변경한다.



js_var_type.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
  <script>
```

```
    let x;           // x의 값은 undefined가 된다.
```

```
    document.write(x + "<br>");
```

```
    document.write(typeof x + "<br><br>");
```

```
    x = 100;         // x은 수치값을 가진다.
```

```
    document.write(x + "<br>");
```

```
    document.write(typeof x + "<br><br>");
```

```
    x = "홍길동";    // x는 문자열을 가진다.
```

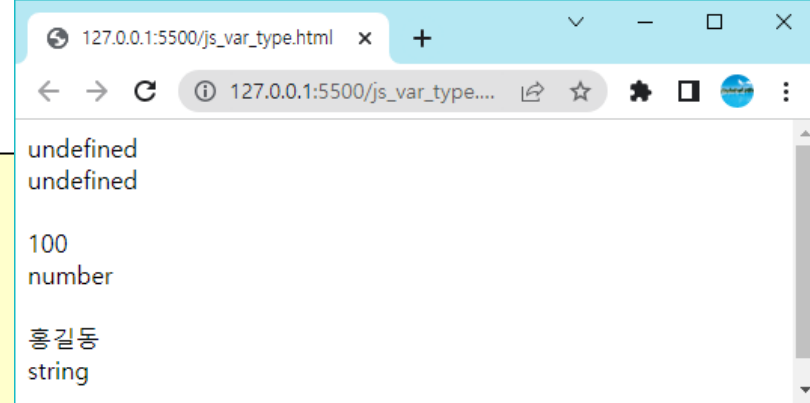
```
    document.write(x + "<br>");
```

```
    document.write(typeof x + "<br><br>");
```

```
  </script>
```

```
</body>
```

```
</html>
```



변수의 사용 범위(scope)

	선언	사용 범위	변수의 생명
전역 변수	함수 밖에서 선언 혹은 var/let 키워드 없이 아무 곳에서도나 선언	프로그램 전역	프로그램이 실행을 시작할 때 생성 프로그램 종료 때 소멸
지역 변수	함수 내에 let으로 선언	선언된 함수 내	함수가 실행될 때 생성 함수가 종료할 때 소멸
블록 변수	let으로 if, while, for 등 블록 내에 선언	선언된 블록 내	블록의 실행 시작 시 생성 블록이 끝나면 소멸

```
let x; // 전역 변수 x 선언
```

```
function f() {
```

```
  let y; // 지역 변수 y 선언
```

```
  x = 10; // 전역 변수 x에 10 저장
```

```
  y = 20; // 지역 변수 y에 20 저장
```

```
  z = 30; // 새로운 전역 변수 z가 선언되고 30 저장됨
```

```
  if(y == 20) {
```

```
    let b = 40; // if 블록에서만 사용되는 블록 변수 b 선언
```

```
    b++;
```

```
  }
```

```
  // 이곳에서는 블록 변수 b에 접근할 수 없음
```

```
  // 이곳에서는 변수 x, y, z에 모두 접근 가능
```

```
}
```

```
// 여기서는 변수 x와 z만 접근 가능. 지역 변수 y와 블록 변수 b 접근 불가
```

지역 변수 y의 사용 범위

블록 변수 b의 사용 범위

전역 변수 x, z의 사용 범위
(프로그램 전체)

this로 전역변수 접근

- 지역 변수와 전역 변수의 이름을 같을 때
 - 전역 변수에 접근하고자 할 때 : **this.전역변수**

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

- 주의
 - let으로 선언된 전역 변수는 this로 접근할 수 없다.

var_scope.html

```
<!DOCTYPE html>
<html>
<head> <meta charset="utf-8"> <title> 변수 선언 </title> </head>
<body>
<h3> 변수 선언, 전역/지역/블록 변수 </h3>
<hr>
<script>
let x;    // 전역 변수 x 선언. var로 선언해도 동일
function f() {
  let y;   // 함수 f() 내에서만 사용되는 지역 변수 y 선언. var로 선언해도 동일
  x = 10;   // 전역 변수 x에 10 저장
  y = 20;   // 지역 변수 y에 20 저장
  z = 30;   // 새로운 전역 변수 z가 선언되고 30이 저장됨
  if(y == 20) {
    let b = 40; // if 블록에서만 사용되는 블록 변수 b 선언
    b++;
    document.write("if 블록 내 블록변수 b = " + b + "<br>");
  }
  // 이곳에서는 블록 변수 b에 접근할 수 없음
  // 이곳에서는 변수 x, y, z에 모두 접근 가능
  document.write("함수 f() 내 지역변수 y = " + y + "<br>");
}

f(); // 함수 f() 호출
document.write("전역변수 x = " + x + "<br>");
document.write("전역변수 z = " + z);
// 이곳에서는 변수 x와 z만 접근 가능, 지역 변수 y와 블록 변수 b는 접근 불가
</script>
</body> </html>
```

변수 선언 - Chrome

localhost/6/ex6-06.html

변수 선언, 전역/지역/블록 변수

if 블록 내 블록변수 b = 41
함수 f() 내 지역변수 y = 20
전역변수 x = 10
전역변수 z = 30

블록 변수 b의 사용 범위

지역 변수 y의 사용 범위

전역 변수 x, z의 사용 범위
(프로그램 전체)

let의 특징

- 도입
 - var를 사용할 때의 코딩 오류(변수 재 선언)를 줄이기 위해
 - 2015년 ES6 표준에 도입
- 특징
 - let으로 동일한 변수 재 선언 불가

```
var x = 1;  
var x = 2; // 정상. 기존 변수 x 제거.  
           // 새로운 변수 x 생성  
           // 개발자의 실수로 코딩 오류 발생
```

```
let x = 1;  
let x = 2; // 오류.  
           //변수 x에 대한 재 선언 불가
```

* var보다 let 사용 권고 – 개발자의 변수 재 선언 실수를 막기 위해

- let은 변수 사용 범위를 블록 내로 제한

```
if(a == b) {  
  let x = 10; // x는 if 블록에서만 사용  
}  
x++; // 오류. x 사용할 수 없음
```

```
for(let n=0; n<10; n++) {  
  let x = 10; // n과 x는 for 블록에서만 사용  
}  
x++; // 오류. x 사용할 수 없음  
n++; // 오류. n 사용할 수 없음
```

상수

- 상수 : 변하지 않는 값을 가지는 이름, **const**로 선언

```
const MAX = 10; // 10의 값을 가지는 상수 MAX 선언
```

- 특징

- 선언된 후 값 수정 불가

```
const MAX = 10;  
MAX = 20; // 오류. 상수는 값을 바꿀 수 없다.
```

- 상수 재선언 불가

```
const MAX = 10;  
...  
const MAX = 10; // 오류. 상수의 재선언 불가
```

- 블록 범위에서만 사용

```
if(a == b) {  
    const MAX = 10;  
    ...  
}  
let m = MAX; // 오류.  
// MAX는 if 블록 밖에서 사용 불가
```

```
for(let n=0; n<10; n++) {  
    const MAX = 10;  
    ...  
}  
let m = MAX; // 오류.  
// MAX는 for 블록 바깥에서 사용 불가
```

자바스크립트의 리터럴

- 리터럴(literal)
 - 데이터 값 그 자체
- 리터럴 종류

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

수치형(number)

- 수치형은 정수 혹은 실수가 가능하다. 실수는 e를 사용하여 지수형으로 표시할 수 있다.

```
let x = 123.00;    // 실수
let y = 123;       // 정수
let z1 = 123e3;    // 123000
let z2 = 123e-3;   // 0.123
```


부울형과 문자열

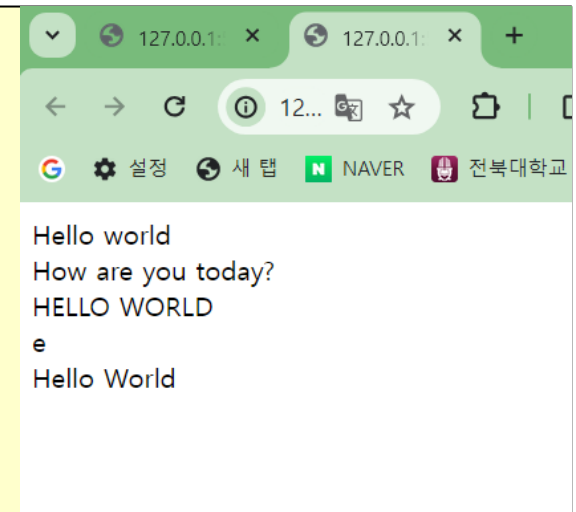
```
let x = (20 > 10);    // x는 true가 된다.  
let y = (10 > 20);    // y는 false가 된다.
```

```
let s1 = "abc";       // s1는 문자열 "abc"를 저장한다.  
let s2 = 'abc';       // OK!  
let s3 = "그는 '슈퍼맨'이라고 불린다"; // 문자열 안에 작은 따옴표를 사용할 수 있다.  
alert(s1.length);     // 3이 출력된다.  
t = "Hello" + "world"; // + 연산자를 사용하여 두개의 문자열을 합친다.
```

- 문자를 그대로 사용하고자 하는 경우 ₩"로 사용할 것

```
let cite="그녀는 ₩"누구세요₩"라고 했습니다.";
```

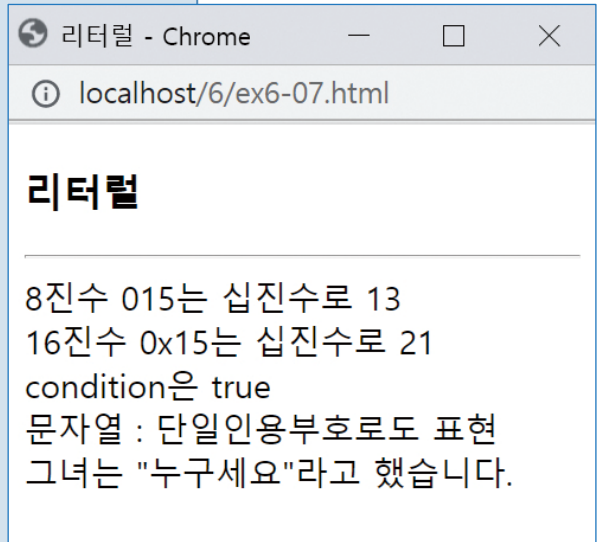
```
<!DOCTYPE html>
<body>
  <script>
    let s = "Hello world";
    let t = "How are you" + " today?";
    document.write(s + "<br>");
    document.write(t + "<br>");
    document.write(s.toUpperCase() + "<br>");
    document.write(s.charAt(1)+ "<br>");
    document.write(s.replace("w", "W"+ "<br>"));
  </script>
</body>
</html>
```



js_literal.html

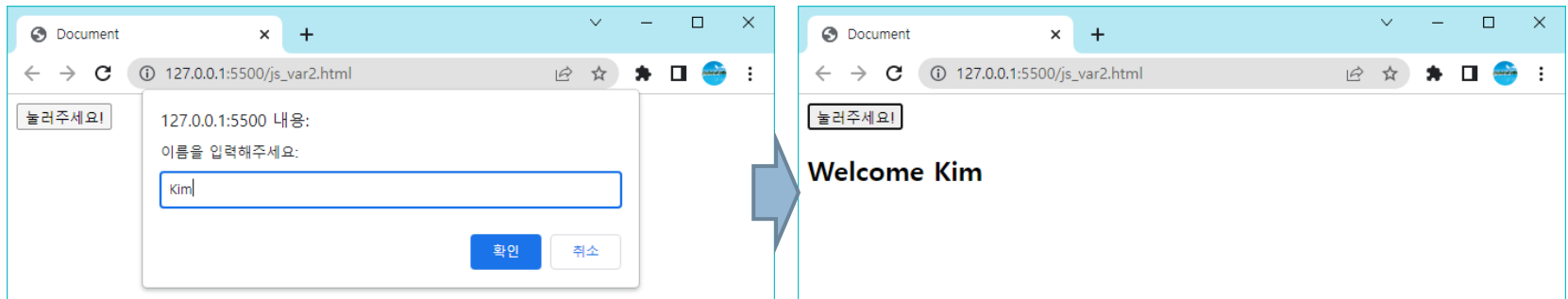
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>리터럴</title> </head>
<body>
<h3>리터럴</h3>
<hr>
<script>
  let oct = 015;    // 015는 8진수. 10진수로 13
  let hex = 0x15; // 0x15는 16진수. 10진수로 21
  let condition = true; // True로 하면 안됨

  document.write("8진수 015는 십진수로 " + oct + "<br>");
  document.write("16진수 0x15는 십진수로 " + hex + "<br>");
  document.write("condition은 " + condition + "<br>");
  document.write('문자열 : 단일인용부호로도 표현' + "<br>");
  document.write("그녀는 ₩"누구세요₩"라고 했습니다.");
</script>
</body>
</html>
```



예: 변수에 문자열 저장하기

- 사용자에게 이름을 입력받아서 환영 문구를 출력하는 프로그램을 작성해 보자.

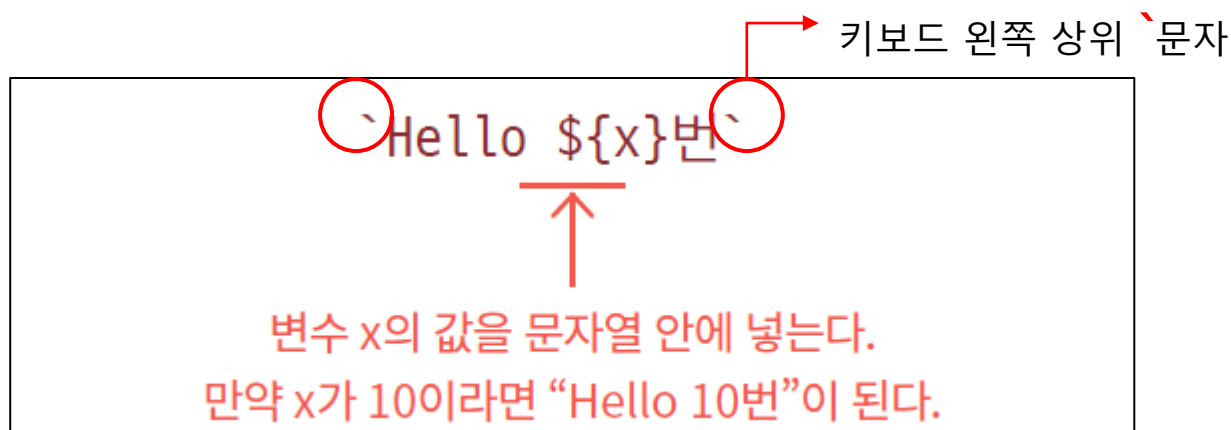


js_var2.html

```
<!DOCTYPE html>
<head>
  <script>
    function sub() {
      const name = prompt('이름을 입력해주세요:');
      document.getElementById("test").innerHTML = "Welcome " +
        name;
    }
  </script>
</head>
<body>
  <button onclick="sub()">눌러주세요!</button>
  <h2 id="test"></h2>
</body>
</html>
```

템플릿 리터럴

- 자바스크립트에서 문자열 안에 변수값을 넣으려면 ``${ 변수 }`` 형태의 문자열을 사용하는 것이 좋다.
- 이 백틱기호(``...``)를 사용한 문자열은 자바스크립트 ES6 버전에서 추가된 것으로 템플릿 리터럴(template literal)이라고 부른다.



```
let x = 10;  
Document.write(`Hello ${x} 번`); // "hello 10번"이 된다.
```

객체형

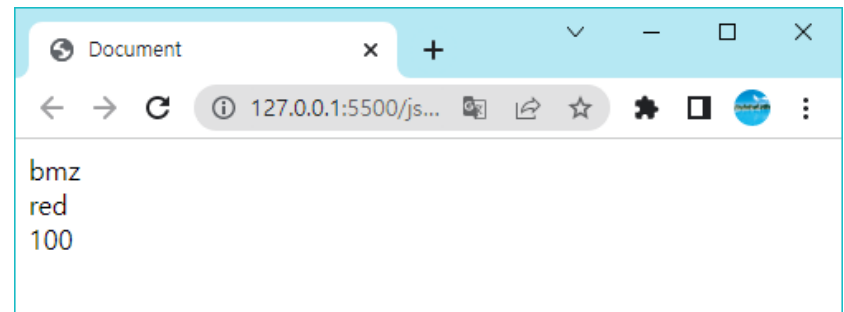
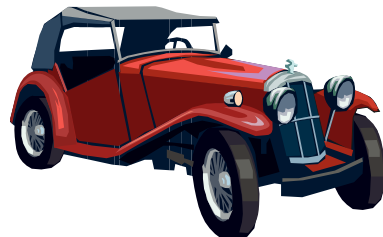
- **객체(object)**는 사물의 속성과 동작을 묶어서 표현하는 기법
- (예) 자동차는 메이커, 모델, 색상, 마력과 같은 속성도 있고 출발하기, 정지하기 등의 동작도 가지고 있다.

```
let myCar = { model: "bmz", color: "red", hp: 100 };
```

```
document.write(myCar.model + "<br>");
```

```
document.write(myCar.color + "<br>");
```

```
document.write(myCar.hp + "<br>");
```



자바스크립트의 식과 연산

- 자바스크립트의 연산과 연산자 종류

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

- 산술 연산자

- 5 가지 : 더하기(+), 빼기(-), 곱하기(*), 나누기(/), 나머지(%)

```
let x = 32;  
let total = 100 + x*2/4 - 3; // total은 113
```

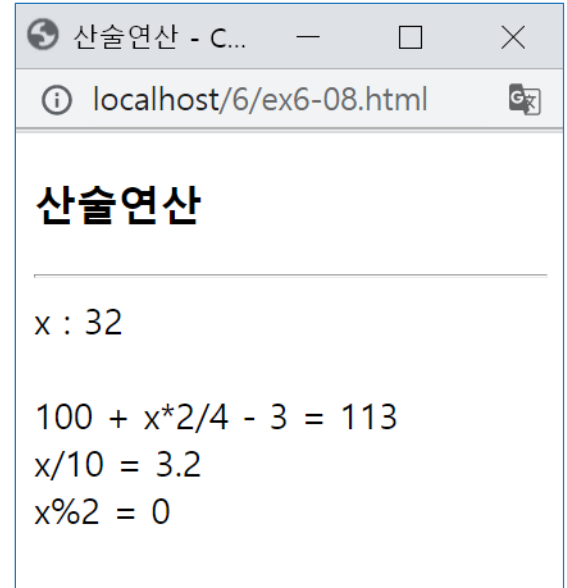
- 연산의 결과는 항상 실수

```
let div = 32/10; // div = 3.2
```


js_op_arith1.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>산술연산</title>
</head>
<body>
<h3>산술연산</h3>
<hr>
<script>
  let x=32;
  let total = 100 + x*2/4 - 3; // total은 113
  let div = x / 10;           // div는 3.2
  let mod = x % 2;           // x를 2로 나눈 나머지, 0

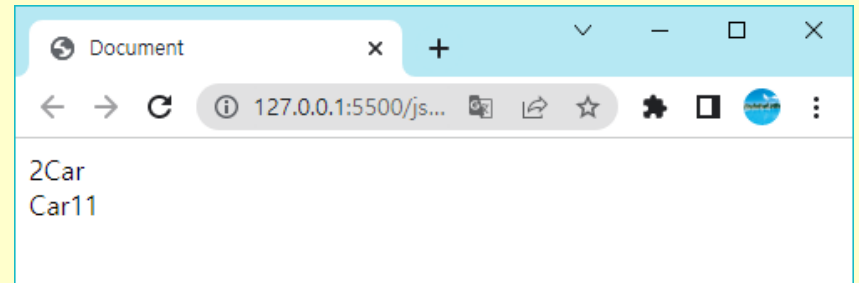
  document.write("x : " + x + "<br><br>");
  document.write("100 + x*2/4 - 3 = " + total + "<br>");
  document.write("x/10 = " + div + "<br>");
  document.write("x%2 = " + mod + "<br>");
</script>
</body>
</html>
```



문자열에서의 + 연산자

js_op_arith2.html

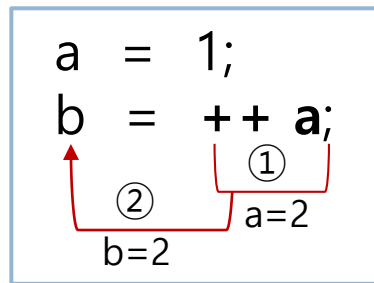
```
<!DOCTYPE html>
<body>
  <script>
    let x, y;
    x = 1 + 1 + "Car";
    y = "Car" + 1 + 1;
    document.write(x + "<br>");
    document.write(y + "<br>");
  </script>
</body>
</html>
```



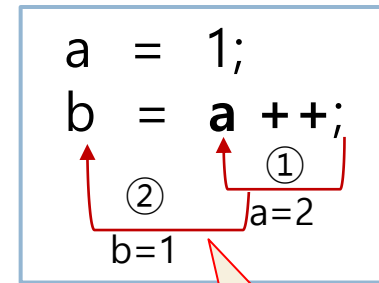
증감 연산자

- 증감 연산자 : ++, --

(a) 전위연산자



(b) 후위연산자



a++의 연산은 증가
전의 값 1을
반환하여 b의 값은
1이 됨

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

대입 연산자

- 대입 연산 : 오른쪽 식의 결과를 왼쪽 변수에 대입

```
let a=1, b=3;  
a = b;      // a에 b의 값이 대입되어 a=3, b=3이 된다.  
a += b;     // a = a + b의 연산이 이루어져, a=6, b=3이 된다.
```

- 대입연산자 종류

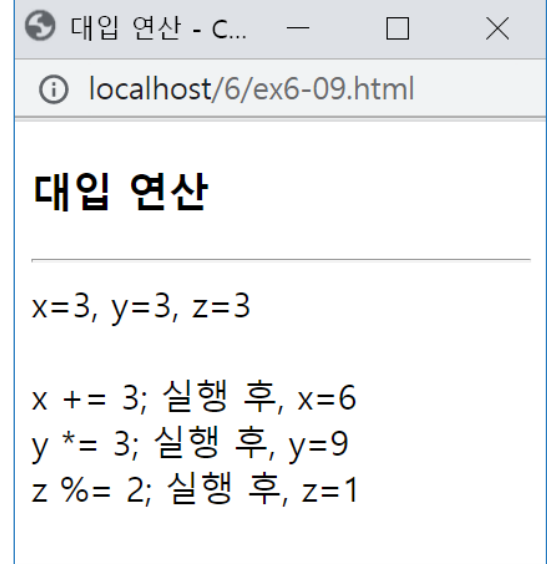
연산자	내용	연산자	내용
a = b	b 값을 a에 대입	a &= b	a = a & b와 동일
a += b	a = a + b와 동일	a ^= b	a = a ^ b와 동일
a -= b	a = a - b와 동일	a = b	a = a b와 동일
a *= b	a = a * b와 동일	a <<= b	a = a << b와 동일
a /= b	a = a / b와 동일	a >>= b	a = a >> b와 동일
a %= b	a = a % b와 동일	a >>>= b	a = a >>> b와 동일

js_op_assignment.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>대입 연산</title>
</head>
<body>
<h3>대입 연산</h3>
<hr>
<script>
  let x=3, y=3, z=3;
  document.write("x=" + x + ", y=" + y);
  document.write(", z=" + z + "<br><br>");

  x += 3;    // x=x+3 -> x=6
  y *= 3;    // y=y*3 -> y=9
  z %= 2;    // z=z%2 -> z=1

  document.write("x += 3; 실행 후, x=" + x + "<br>");
  document.write("y *= 3; 실행 후, y=" + y + "<br>");
  document.write("z %= 2; 실행 후, z=" + z);
</script>
</body>
</html>
```



비교 연산자

- 비교 연산 : 두 값 비교, true나 false의 결과를 내는 연산

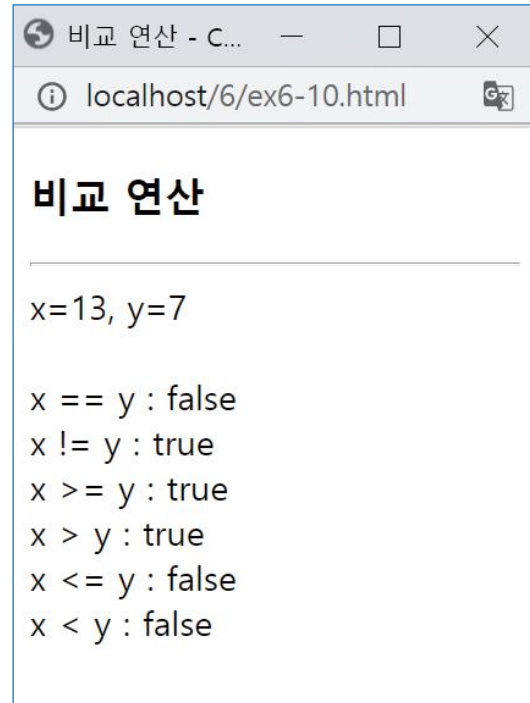
```
let age = 25;  
let result = (age > 20);    // age가 20보다 크므로 result는 true
```

- 비교 연산자 종류

연산자	내용	연산자	내용
$a < b$	a가 b보다 작으면 true	$a \geq b$	a가 b보다 크거나 같으면 true
$a > b$	a가 b보다 크면 true	$a == b$	a가 b와 같으면 true
$a \leq b$	a가 b보다 작거나 같으면 true	$a \neq b$	a가 b와 같지 않으면 true

js_op_compa.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>비교 연산</title>
</head>
<body>
<h3>비교 연산</h3>
<hr>
<script>
  let x=13, y=7;
  document.write("x=" + x + ", y=" + y + "<br><br>");
  document.write("x == y : " + (x == y) + "<br>");
  document.write("x != y : " + (x != y) + "<br>");
  document.write("x >= y : " + (x >= y) + "<br>");
  document.write("x > y : " + (x > y) + "<br>");
  document.write("x <= y : " + (x <= y) + "<br>");
  document.write("x < y : " + (x < y) + "<br>");
</script>
</body>
</html>
```



논리 연산자

- 논리 연산 : AND, OR, NOT

```
let score = 90;  
let age = 20;  
let res = ((score > 80) && (age < 25)); // res=true
```

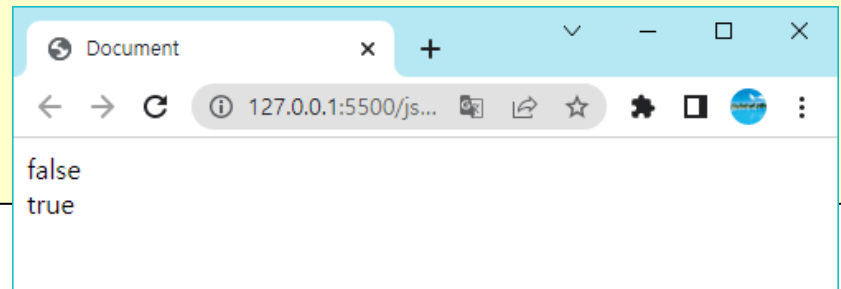
- 논리 연산 종류

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

논리연산자

js_op_logic.html

```
<!DOCTYPE html>
<body>
  <script>
    let x = 3;
    let y = 4;
    document.write((x == 3 && y == 7) + "<br>");
    document.write((x == 3 || y == 4) + "<br>");
  </script>
</body>
</html>
```



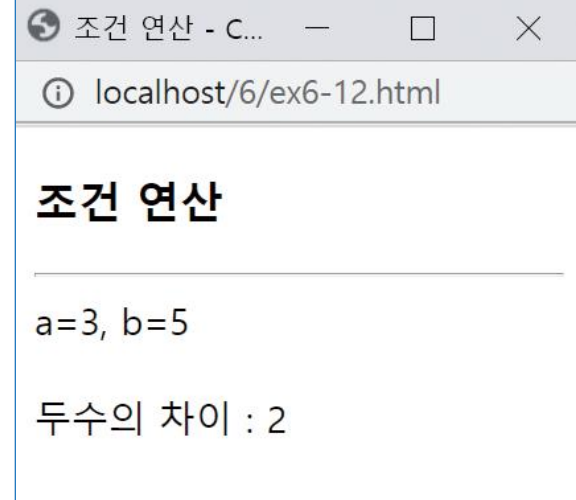
조건 연산자

- 조건 연산
 - `condition ? expT : expF`
 - `condition`이 `true`이면 전체 결과는 `expT`의 계산 값
 - `false`이면 `expF`의 계산 값

```
let x=5, y=3;  
let big = (x>y) ? x : y; // (x>y)가 true이므로 x 값 5가 big에 대입된다.
```

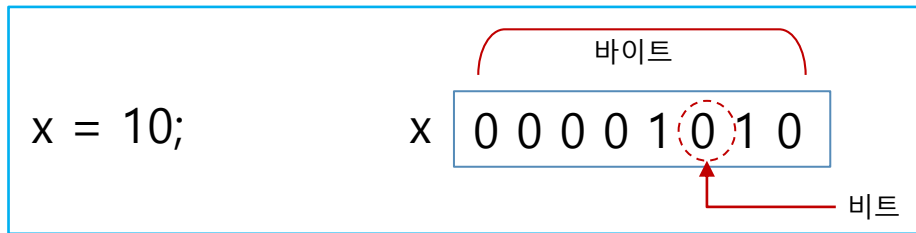
js_op_conditional.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>조건 연산</title>
</head>
<body>
<h3>조건 연산</h3>
<hr>
<script>
  let a=3, b=5;
  document.write("a=" + a + ", b=" + b + "<br><br>");
  document.write("두수의 차이 : " + ((a>b)?(a-b):(b-a)));
</script>
</body>
</html>
```



비트 연산

- 비트 개념



- 비트 연산 종류

- 비트들끼리의 비트 논리 연산
- 비트 시프트 연산

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
$a \wedge b$	비트 XOR 연산	두 비트가 다르면 1, 같으면 0
$\sim a$	비트NOT 연산	1을 0으로, 0을 1로 변환

$a = 106;$ `0 1 1 0 1 0 1 0`

$b = 77;$ `0 1 0 0 1 1 0 1`

$c = a \& b;$

```
  0 1 1 0 1 0 1 0
& 0 1 1 0 1 1 0 1
-----
c  0 1 1 0 1 0 0 0
```

둘 다 1,
결과 1

하나라도 0,
결과 0

$c = a | b;$

```
  0 1 1 0 1 0 1 0
| 0 1 0 0 1 1 0 1
-----
c  0 1 1 0 1 1 1 1
```

둘 다 0,
결과 1

하나라도 1,
결과 1

$c = a \wedge b;$

```
  0 1 1 0 1 0 1 0
^ 0 1 0 0 1 1 0 1
-----
c  0 0 1 0 0 1 1 1
```

둘이 같으면
결과 0

둘이 다르면,
결과 1

$c = \sim a;$

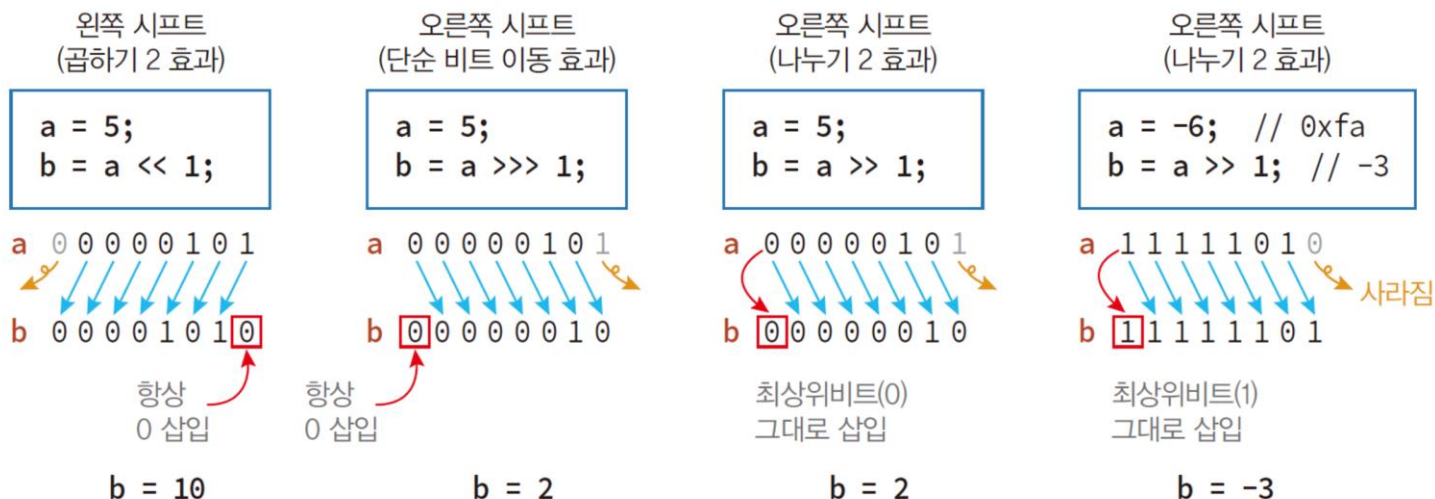
```
~ 0 1 1 0 1 0 1 0
c 1 0 0 1 0 1 0 1
```

1은 0으로
바꿈

0은 1로
바꿈

비트 시프트 연산

- 시프트 : 저장 공간에서 비트들의 오른쪽/왼쪽 이동



연산자	별칭	설명
$a \ll b$	산술적 왼쪽 시프트	a 의 비트들을 왼쪽으로 b 번 이동. 최하위 비트의 빈자리는 0으로 채움. 한 비트 시프트마다 곱하기 2의 효과 발생. a 값은 변화 없음
$a \gg b$	산술적 오른쪽 시프트	a 의 비트들을 오른쪽으로 b 번 이동. 최상위 비트의 빈자리는 시프트 전 최상위 비트로 채움. 한 비트 시프트마다 나누기 2의 효과 발생. a 값은 변화 없음
$a \ggg b$	논리적 오른쪽 시프트	a 의 비트들을 오른쪽으로 b 번 이동. 최상위 비트의 빈자리는 0으로 채움. a 값은 변화 없음

문자열 연산자

- 문자열 연결

- + , +=

"abc" + "de"	// "abcde"
"abc" + 23	// "abc23"
23 + "abc"	// "23abc"
23 + "35"	// "2335"
23 + 35	// 58, 정수 더하기

- 순서에 유의

23 + 35 + "abc";	// 23 + 35 -> 58로 먼저 계산, 58 + "abc" -> "58abc"
"abc" + 23 + 35;	// "abc" + 23 -> "abc23"로 먼저 계산, "abc23" + 35 -> "abc2335"

- 문자열 비교

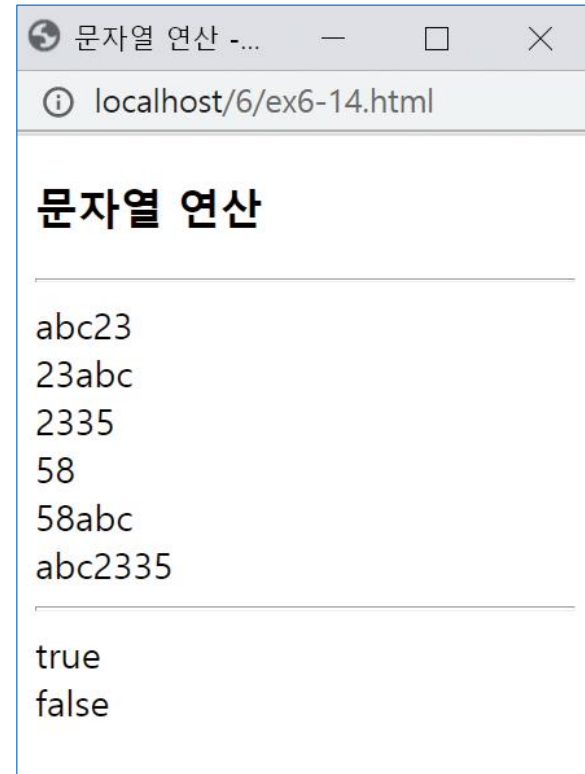
- 비교 연산자(!=, ==, > , <, <=, >=)는 문자열 비교에 사용
 - 사전 순으로 비교 결과 리턴

```
let name = "kitae";  
let res = (name == "kitae"); // 비교 결과 true, res = true  
let res = (name > "park"); // name이 "park"보다 사전순으로 앞에 나오므로 res = false
```

js_op_string.html

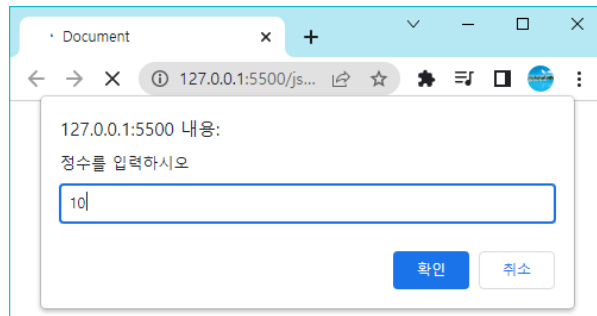
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>문자열 연산</title>
</head>
<body>
<h3>문자열 연산</h3>
<hr>
<script>
  document.write("abc" + 23 + "<br>");
  document.write(23 + "abc" + "<br>");
  document.write(23 + "35" + "<br>");
  document.write(23 + 35 + "<br>");
  document.write(23 + 35 + "abc" + "<br>");
  document.write("abc" + 23 + 35 + "<br><hr>");

  let name = "kitae";
  document.write(name == "kitae");
  document.write("<br>");
  document.write(name > "park");
</script>
</body>
</html>
```



예제: 덧셈 프로그램

- 예를 들어서 사용자로 부터 2개의 숫자를 받아서 덧셈을 하고 그 결과를 화면에 표시하는 자바스크립트 프로그램을 작성하여 보자.



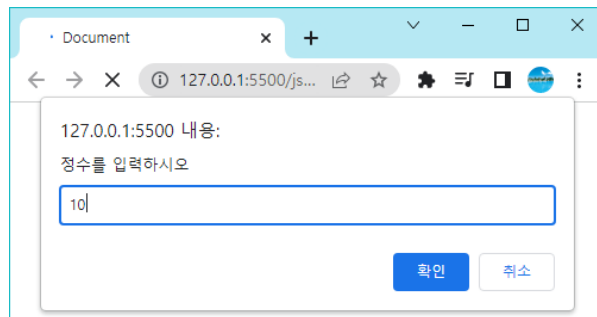
Document x + - □ ×

127.0.0.1:5500/js... ☆

127.0.0.1:5500 내용:
정수를 입력하십시오

10

확인 취소



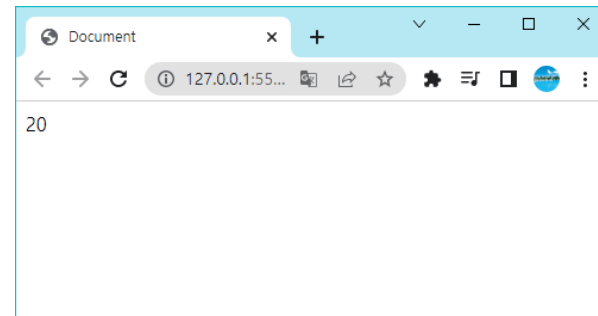
Document x + - □ ×

127.0.0.1:5500/js... ☆

127.0.0.1:5500 내용:
정수를 입력하십시오

10

확인 취소



Document x + - □ ×

127.0.0.1:55... ☆

20

예제:

js_parse.html

```
<script>
  let x, y;
  let input;

  input = prompt("정수를 입력하십시오", "정수로");
  x = parseInt(input);

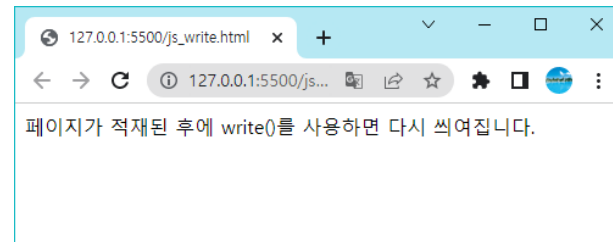
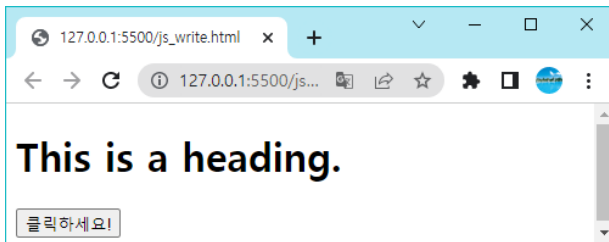
  input = prompt("정수를 입력하십시오", "정수로");
  y = parseInt(input);
  document.write(x+y + "<br>");
</script>
```

HTML 문서에 쓰기

- 자바스크립트의 HTML 문서의 요소들을 읽을 수 있어야 하고, HTML 문서에 요소를 출력할 수 있어야 한다. HTML 문서에 출력하는 방법과 특정 HTML 요소에 접근하는 방법을 살펴보자.
- 페이지가 로딩된 후에 `document.write()`를 호출한다면 전체 HTML 페이지가 다시 씌여진다. 즉 이전에 만들어졌던 요소들이 전부 삭제된다.

js_write2.html

```
<!DOCTYPE html>
<html>
<body>
  <h1 id="test">This is a heading.</h1>
  <script>
    function func() {
      document.write("페이지가 적재된 후에 write()를 사용하면 다시 씩여집니다.");
    }
  </script>
  <button type="button" onclick="func()">클릭하세요!</button>
</body>
</html>
```

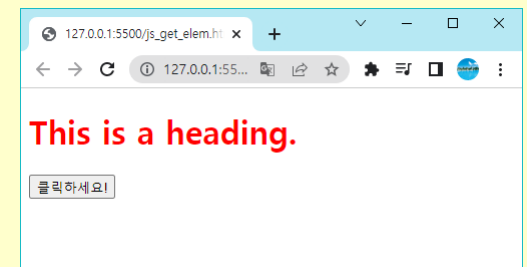
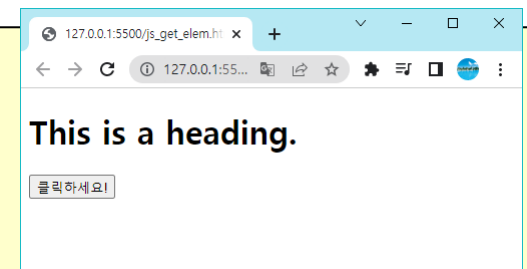


HTML 요소에 접근하기

- 자바스크립트에서 HTML 요소에 접근하기 위해서는 `document.getElementById(id)` 메소드를 사용한다. HTML 요소를 식별하기 위해서 "id" 속성을 사용한다.

js_get_elem.html

```
<!DOCTYPE html>
<html>
<body>
  <h1 id="test">This is a heading.</h1>
  <script>
    function func() {
      e = document.getElementById("test");
      e.style.color = "red";
    }
  </script>
  <button type="button" onclick="func()">클릭하세요!</button>
</body>
</html>
```



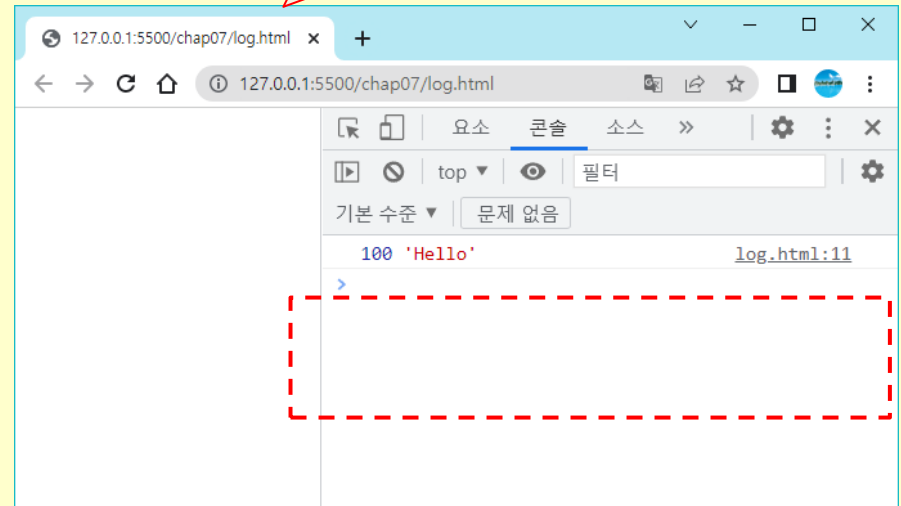
console.log()

- 자바스크립트 코드를 디버깅할 때 많이 사용하는 함수가 console.log()이다.
- 수식의 값을 웹브라우저의 콘솔 창에 출력하는 함수

console.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
</head>
<body>
  <script>
    let x = 100;
    let y = 'Hello';
    console.log(x, y);
  </script>
</body>
</html>
```

Ctrl+Shift+J를 눌러 콘솔창을
열어볼수있다.
또는 웹브라우저에서
마우스오른쪽버튼누른뒤 "**검사**"



if, if-else

● if, if-else 문

```
if(조건식) {  
    ... 실행문 ... // 조건식이 참인 경우  
}
```

```
if(a > b) {  
    document.write("a가 크다");  
}
```

```
if(조건식) {  
    ... 실행문1 ... // 조건식이 참인 경우  
}  
else {  
    ... 실행문2 ... // 조건식이 거짓인 경우  
}
```

```
if(a > b) {  
    document.write("a가 크다");  
}  
else {  
    document.write("a가 크지 않다");  
}
```

```
if(조건식1) {  
    실행문1 // 조건식1이 참인 경우  
}  
else if(조건식2) {  
    실행문2 // 조건식2가 참인 경우  
}  
.....  
else {  
    실행문n; // 앞의 모든 조건이 거짓인 경우  
}
```

```
if(a > b) {  
    document.write("a가 크다");  
}  
else if(a < b) {  
    document.write("b가 크다");  
}  
else  
    document.write("a와 b는 같다");
```

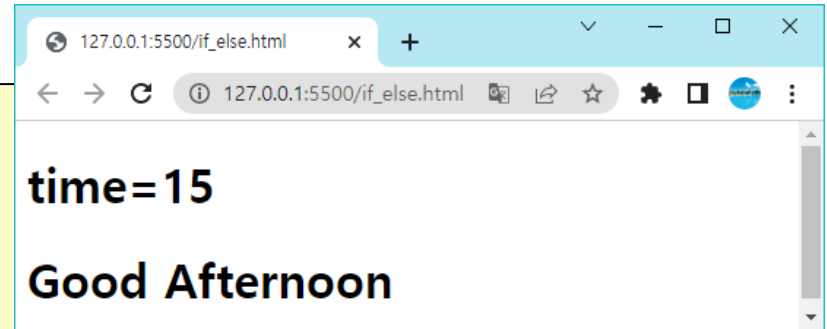
if-else 문

if-else.html

```
<!DOCTYPE html>
<html>
<body>
  <h1 id="test1"></h1>
  <h1 id="test2"></h1>
  <script>
    let time = new Date().getHours();
    let msg;
    document.getElementById("test1").innerHTML = "time="+time;

    if (time < 12) {
      msg = "Good Morning";
    } else {
      msg = "Good Afternoon";
    }

    document.getElementById("test2").innerHTML = msg;
  </script>
</body>
</html>
```



if-elseif.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>if-else</title>
</head>
<body>
<h3>if-else를 이용한 학점 매기기</h3>
<hr>
<script>
  let grade;
  let score = prompt("황기태 님 점수를 입력하세요", 100);
  score = parseInt(score); // 문자열을 숫자로 바꿈
  if(score >= 90) // score가 90 이상
    grade = "A";
  else if(score >= 80) // 80 이상 90 미만
    grade = "B";
  else if(score >= 70) // 70 이상 80 미만
    grade = "C";
  else if(score >= 60) // 60 이상 70 미만
    grade = "D";
  else // 60 미만
    grade = "F";
  document.write(score + "는 " + grade + "입니다.<br>")
</script>
</body>
</html>
```

localhost 내용:

황기태 님 점수를 입력하세요

확인

취소

if-else - Chrome

localhost/6/ex6-15.html

if-else를 이용한 학점 매기기

95는 A입니다.

switch 문

- switch 문
 - 값에 따라 서로 다른 코드를 실행할 때, switch문 적합

```
switch(식) {  
  case 값1: // 식의 결과가 값1과 같을 때  
    실행 문장 1;  
    break;  
  case 값2: // 식의 결과가 값2와 같을 때  
    실행 문장 2;  
    break;  
  ...  
  case 값m:  
    실행 문장 m; // 식의 결과가 값과 같을 때  
    break;  
  default: // 어느 값과도 같지 않을 때  
    실행 문장 n;  
}
```

```
let fruits="사과";  
switch(fruits) {  
  case "바나나":  
    price = 200; break;  
  case "사과":  
    price = 300; break;  
  case "체리":  
    price = 400; break;  
  default:  
    document.write("팔지 않습니다.");  
    price = 0;  
}  
  
// switch 문의 실행 결과 price=300
```

case 문의 '값'

- case 문의 '값'은 const로 선언된 상수나 리터럴만 가능
 - 잘 작성된 case 문

```
const MAX = 100;  
...  
...  
case 1 :  
case 2.7 :  
case "Seoul" :  
case MAX :  
case true :  
case 2+3 : // 2+3은 먼저 5로 계산되어 case 5:와 동일
```

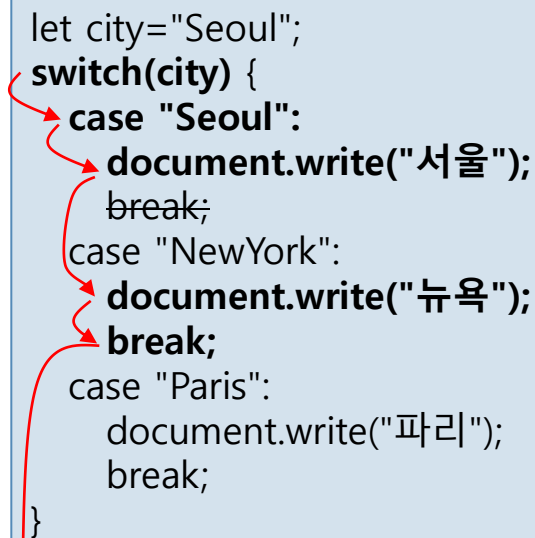
- case 문의 '값'에 변수나 식은 사용 불가
 - 잘못 작성된 case 문

```
case a :           // 오류. 변수 a 사용 불가  
case a > 3 :       // 오류. 식(a > 3) 사용 불가
```

switch 문에서 break 문의 역할

- break 문
 - switch 문 종료
 - break; 문을 만날 때까지 아래로 코드 계속 실행

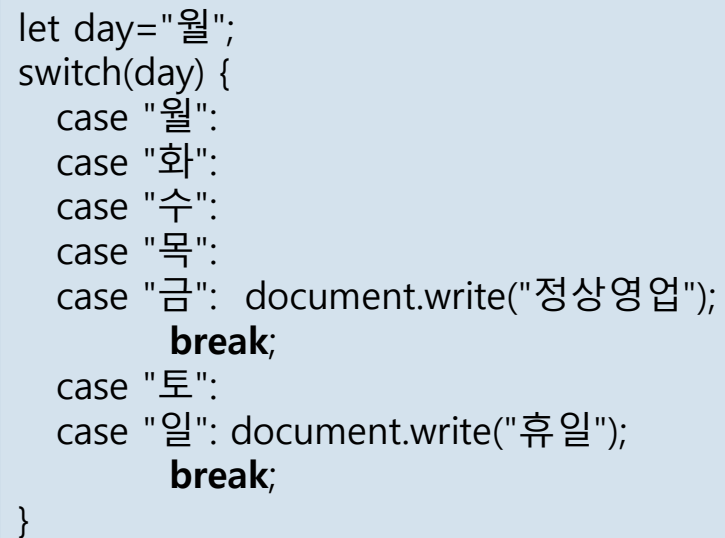
```
let city="Seoul";  
switch(city) {  
  case "Seoul":  
    document.write("서울");  
    break;  
  case "NewYork":  
    document.write("뉴욕");  
    break;  
  case "Paris":  
    document.write("파리");  
    break;  
}
```



서울뉴욕

(a) break;를 만날 때까지 아래로 실행을 계속하는 사례

```
let day="월";  
switch(day) {  
  case "월":  
  case "화":  
  case "수":  
  case "목":  
  case "금": document.write("정상영업");  
    break;  
  case "토":  
  case "일": document.write("휴일");  
    break;  
}
```



정상영업

(b) 여러 case에 대해 동일한 코드를 실행하도록 의도적으로 break; 를 생략한 경우

js_switch1.html

```
<!DOCTYPE html>
<html>
<body>
  <h1 id="test"></h1>
  <script>
```

```
    let day = new Date().getDay();
```

```
    switch (day) {
```

```
      case 0:
```

```
      case 6:
```

```
        day = "주말";
```

```
        break;
```

```
      case 1:
```

```
      case 2:
```

```
      case 3:
```

```
      case 4:
```

```
      case 5:
```

```
        day = "주중";
```

```
        break;
```

```
    }
```

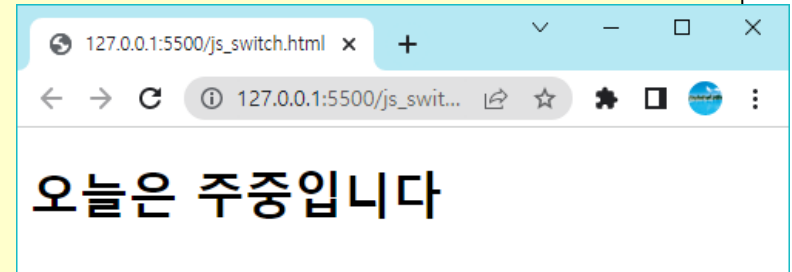
```
    document.getElementById("test").innerHTML = "오늘은 " + day + "입니다";
```

```
  </script>
```

```
</body>
```

```
</html>
```

```
getFullYear(); // 2024
getMonth();    // 월을 출력(0-11)
getDate();     // 날짜를 출력(1-31)
getDay();      // 요일을 출력(0-6), 월요일->1
getHours(); getMinutes(), getSeconds(), getTime;
```



js_switch2.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>switch</title>
</head>
<body>
<h3>switch 문으로 커피 주문</h3>
<hr>
<script>
  let price = 0;
  let coffee = prompt("무슨 커피 드릴까요?", "");
  switch(coffee) {
    case "espresso" :
    case "에스프레소" : price = 2000;
      break;
    case "카푸치노" : price = 3000;
      break;
    case "카페라떼" : price = 3500;
      break;
    default :
      document.write(coffee + "는 없습니다.");
  }
  if(price != 0)
    document.write(coffee + "는 " + price + "원입니다.");
</script>
</body>
</html>
```

"espresso"나
"에스프레소"의 경우
모두 실행

localhost 내용:

무슨 커피 드릴까요?

espresso

확인

취소

switch - Chrome

localhost/6/ex6-16.html

switch 문으로 커피 주문

espresso는 2000원입니다.

반복문

- for 문

```
for(초기문; 조건식; 반복 후 작업) {  
    ... 작업문 ...  
}
```

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

0123456789

- while 문

```
while(조건식) {  
    ... 작업문 ...  
}
```

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

0123456789

- do-while 문

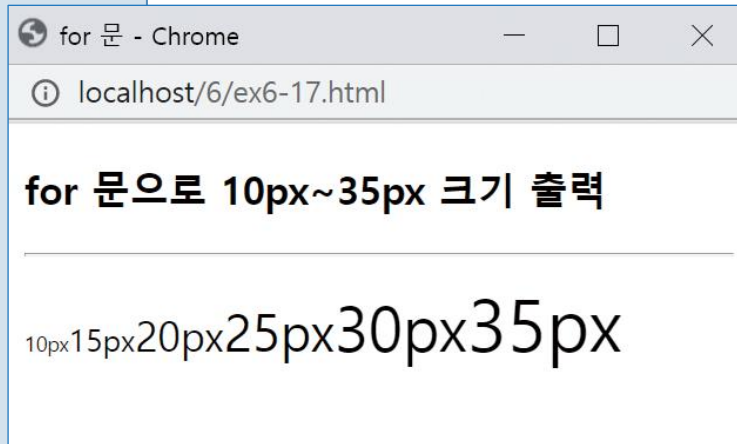
```
do {  
    ... 작업문 ...  
} while(조건식);
```

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

0123456789

js_for.html

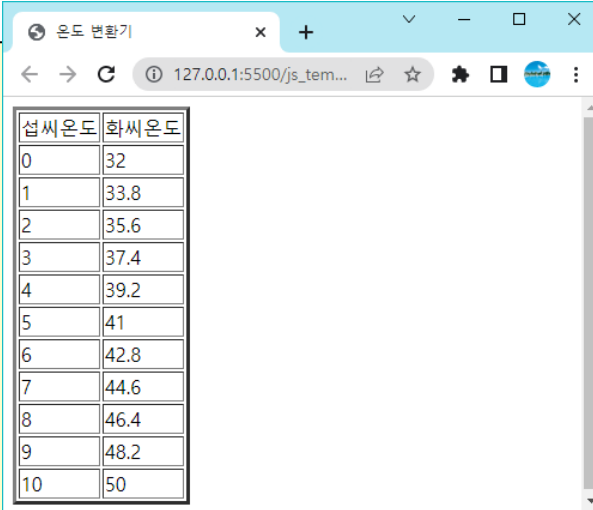
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>for 문</title>
</head>
<body>
<h3>for 문으로 10px~35px 크기 출력</h3>
<hr>
<script>
  for(let size=10; size<=35; size+=5) { // 5씩 증가
    document.write("<span ");
    document.write("style='font-size:" + size + "px'>");
    document.write(size + "px");
    document.write("</span>");
  }
</script>
</body>
</html>
```



js_temp.html

```
<html>
<head>
  <title>온도 변환기</title>
</head>

<body>
  <table border="3">
    <tr>
      <td>섭씨온도</td>
      <td>화씨온도</td>
    </tr>
    <script>
      for (celsius = 0; celsius <= 10; celsius = celsius + 1) {
        document.write("<tr><td>" + celsius + "</td><td>"
          + ((celsius * 9.0 / 5) + 32) + "</td></tr>");
      }
    </script>
  </table>
</body>
</html>
```



섭씨온도	화씨온도
0	32
1	33.8
2	35.6
3	37.4
4	39.2
5	41
6	42.8
7	44.6
8	46.4
9	48.2
10	50

js_nested_loop.html

```
<script>
```

```
document.write("<h1>구구단표</h1>");  
document.write("<table border=2 width=50%");
```

```
for (let i = 1; i <= 9; i++) {  
    document.write("<tr>");  
    document.write("<td>" + i + "</td>");  
  
    for (let j = 2; j <= 9; j++) {  
        document.write("<td>" + i * j + "</td>");  
    }  
    document.write("</tr>");  
}  
document.write("</table>");
```

```
</script>
```

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

js_while.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>while 문</title>
</head>
<body>
<h3>while 문으로 0에서 n까지 합</h3>
<hr>
<script>
    let n = prompt("0보다 큰 정수를 입력하세요", 0);
    n = parseInt(n); // 문자열 n을 숫자로 바꿈

    let i=0, sum=0;
    while(i<=n) { // i가 0에서 n까지 반복
        sum += i;
        i++;
    }
    document.write("0에서 " + n + "까지 합은 " + sum);
</script>
</body>
</html>
```

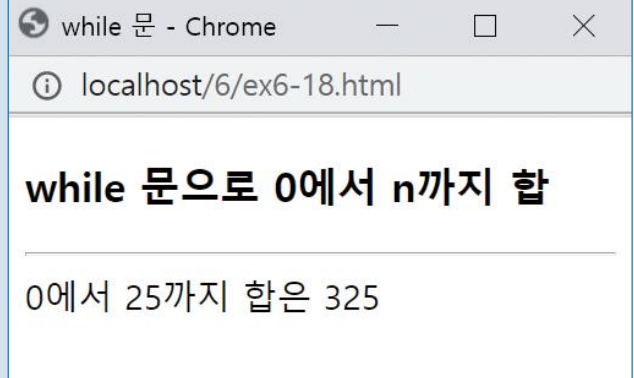
prompt()가 리턴한 것은 문자열

localhost 내용:

0보다 큰 정수를 입력하세요

확인

취소



js_do_while.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>do-while 문</title>
</head>
<body>
<h3>do-while 문으로 0에서 n까지 합</h3>
<hr>
<script>
    let n = prompt("0보다 큰 정수를 입력하세요", 0);
    n = parseInt(n); // 문자열 n을 숫자로 바꿈

    let i=0, sum=0;
    do {
        sum += i;
        i++;
    } while(i<=n); // i가 0~n까지 반복
    document.write("0에서 " + n + "까지 합은 " + sum);
</script>
</body>
</html>
```

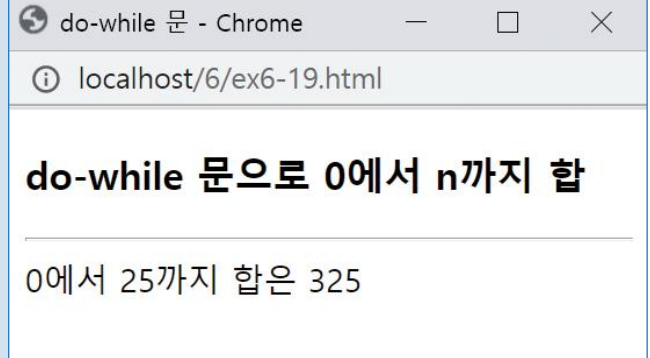
prompt()가 리턴한 것은 문자열

localhost 내용:

0보다 큰 정수를 입력하세요

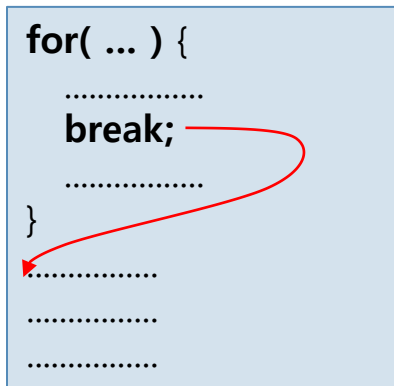
확인

취소

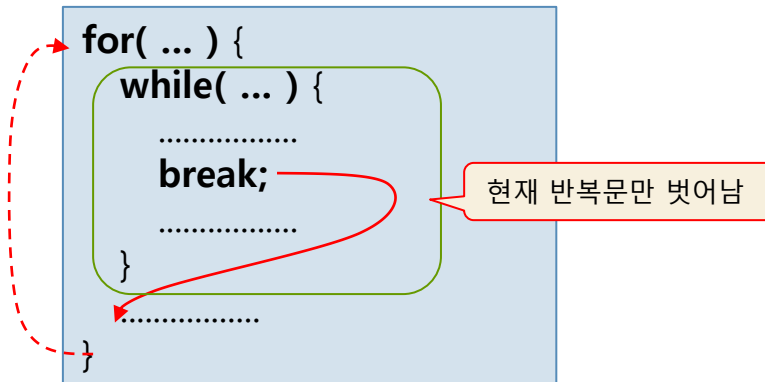


반복문 내의 break 문과 continue 문

- break 문 : 가장 안쪽 반복문 하나만 벗어나도록 제어

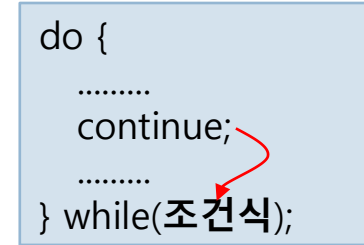
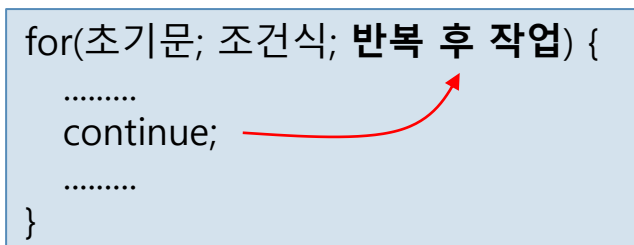


(a) 반복문 벗어나기



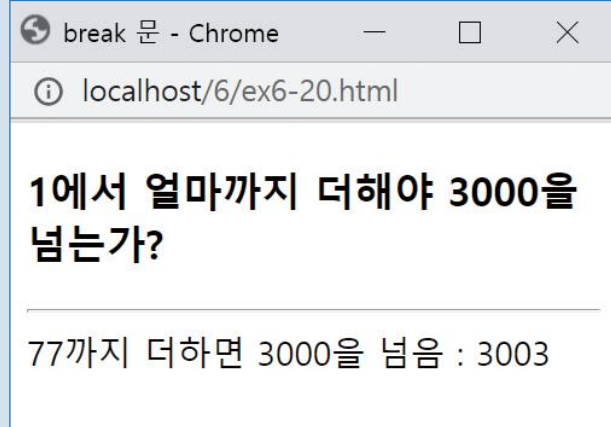
(b) 중첩 반복에서 현재 반복문만 벗어남

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



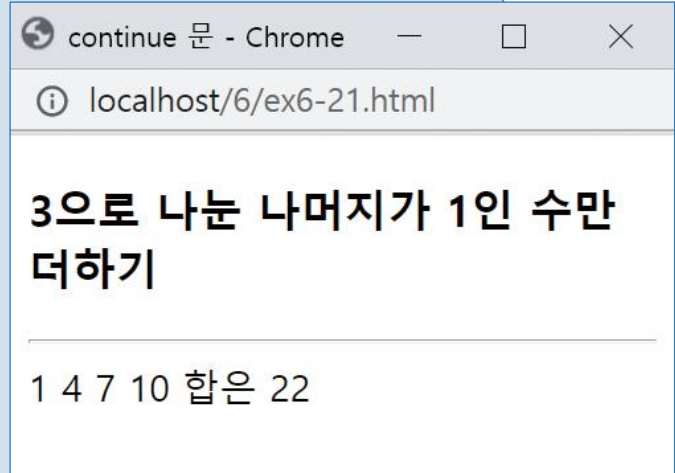
js_while_break.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>break 문</title>
</head>
<body>
<h3>1에서 얼마까지 더해야 3000을 넘는가?</h3>
<hr>
<script>
  let i=0, sum=0;
  while(true) { // 무한 반복
    sum += i;
    if(sum > 3000)
      break; // 합이 3000보다 큼. 반복문 벗어남
    i++;
  }
  document.write(i + "까지 더하면 3000을 넘음 : " + sum);
</script>
</body>
</html>
```



js_for_continue.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>continue 문</title>
</head>
<body>
<h3>3으로 나눈 나머지가 1인 수만 더하기</h3>
<hr>
<script>
  let sum=0;
  for(let i=1; i<=10; i++) { // i가 1에서 10까지 반복
    if(i%3 != 1)              // 3으로 나눈 나머지가 1이 아닌 경우
      continue;              // 다음 반복으로 점프(i++ 코드로)
    document.write(i + " ");
    sum += i;
  }
  document.write("합은 " + sum);
</script>
</body>
</html>
```



함수

- 함수의 구성

```
function 함수이름(arg1, arg2,..., argn) {  
    ...프로그램 코드...  
    결과를 리턴하는 return 문  
}
```

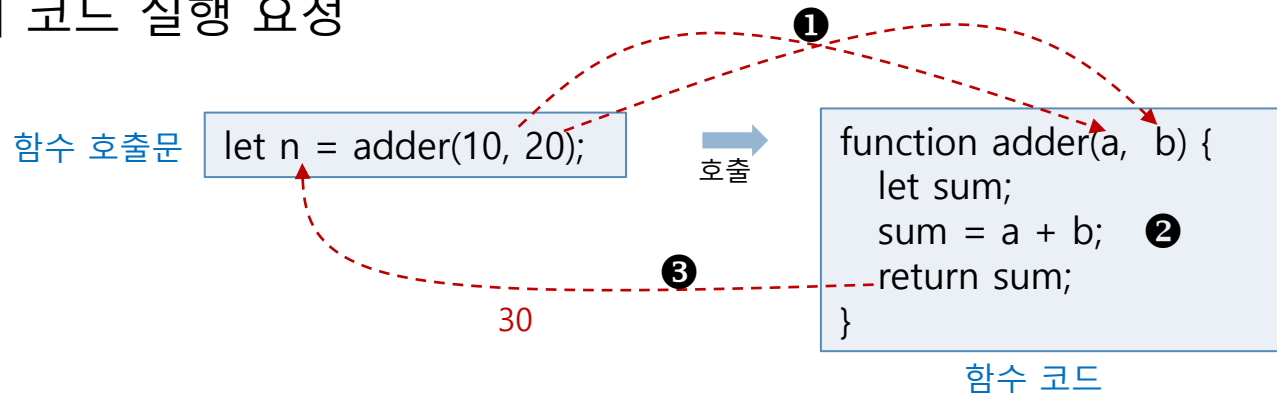
함수 선언 함수 이름 매개 변수

```
function adder ( a, b ) {  
    let sum;  
    sum = a + b;  
    return sum; // 덧셈 합 리턴  
}
```

반환 키워드 반환 값

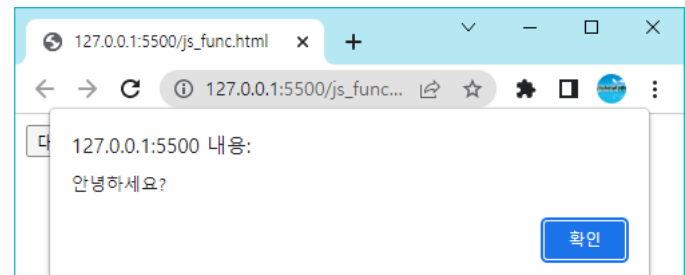
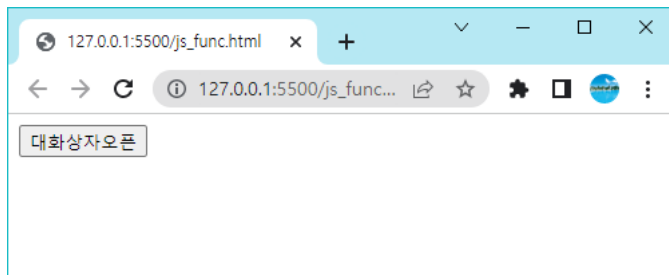
- 함수 호출

- 함수의 코드 실행 요청



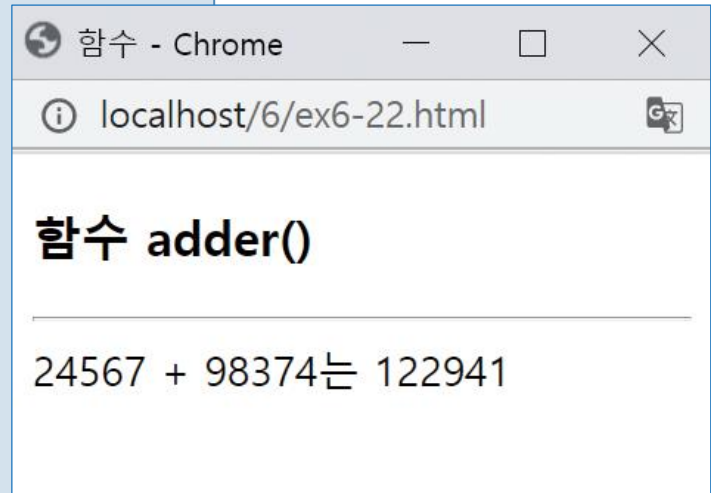
js_func1.html

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function showDialog() {
      alert("안녕하세요?");
    }
  </script>
</head>
<body>
  <button onclick="showDialog()">대화상자오픈</button>
</body>
</html>
```



js_func2.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>함수</title>
<script>
function adder(a, b) { // 함수 작성
    let sum;
    sum = a + b;
    return sum;
}
</script>
</head>
<body>
<h3>함수 adder()</h3>
<hr>
<script>
    let n = adder(24567, 98374); // 함수 호출
    document.write("24567 + 98374는 " + n + "<br>");
</script>
</body>
</html>
```



무명함수

- 자바스크립트에서는 함수 이름을 주지 않고 만들어서 한 번만 사용하는 경우도 많다. 이것을 무명 함수(anonymous function)라고 한다.

```
function showDialog() {  
    alert("안녕하세요?");  
}
```

->

```
let greeting = function()  
{  
    alert("안녕하세요?");  
};  
greeting();
```

화살표 함수

- 화살표 함수(arrow function)는 ES6에 도입되었다. 화살표 함수를 사용하면 더 짧은 함수 구문을 작성할 수 있다. 화살표 함수는 자바 언어의 람다식과 유사하다

```
function sub(a, b){  
  return a + b;  
}
```



```
(a, b) => { return a+b; }  
또는  
(a, b) => return a+b;  
또는  
(a, b) => a + b;
```

```
function sub(a, b){  
  let data = 29;  
  return a + b + data;  
}
```

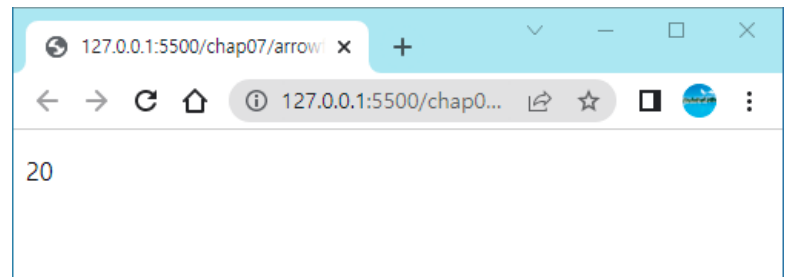


```
(a, b) => {  
  let data = 29;  
  return a + b + data;  
}
```

예제

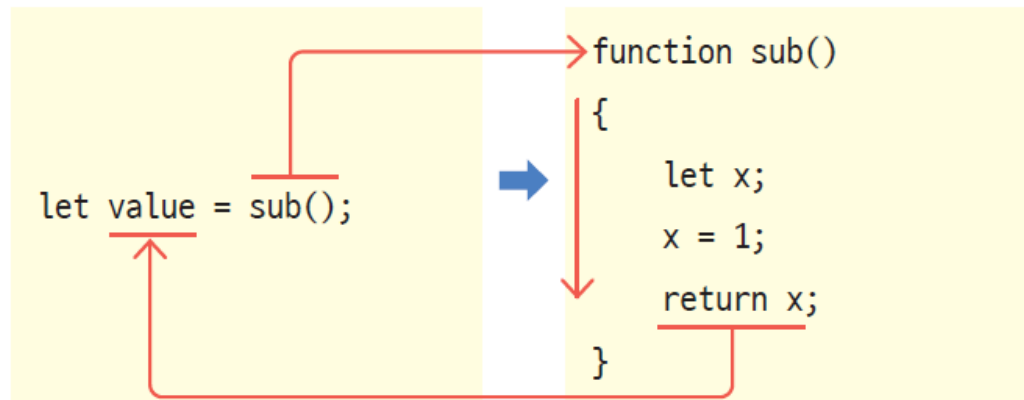
arrow_func.html

```
<!DOCTYPE html>
<html>
<body>
  <p id="test"></p>
  <script>
    let func = (a, b) => a * b;
    document.getElementById("test").innerHTML = func(4, 5);
  </script>
</body>
</html>
```



함수의 반환값

- return 문장을 사용하여 외부로 값을 반환



자바스크립트에서 제공하는 전역 함수

- 대표적인 자바스크립트 함수

- eval() 함수

- 예) `let res = eval("2*3+4*6");` // res는 30

- parseInt() 함수

- 예) `let l = parseInt("32");` // "32"를 10진수로 변환, 정수 32 리턴

- `let n = parseInt("0x32");` // "0x32"를 16진수로 해석, 정수 50 리턴

- isNaN() 함수

- 예) `isNaN(32)` // false 리턴. 숫자이므로

- `isNaN('32')` // false 리턴. 숫자 값의 문자열이므로

- `isNaN("32")` // false 리턴. 숫자 값의 문자열이므로

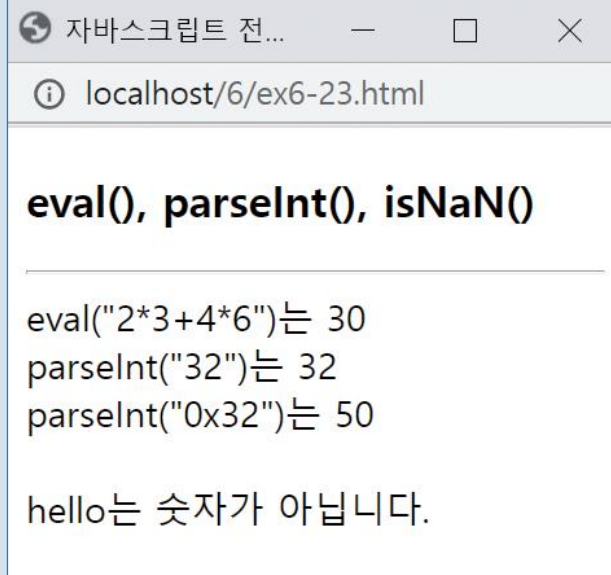
- `isNaN("hello")` // true 리턴. 숫자가 아니므로

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

js_global_func1.html

```
<!DOCTYPE html>
<html>
<head><meta charset="utf-8">
<title>자바스크립트 전역함수</title>
<script>
function evalParseIntIsNaN() {
    let res = eval("2*3+4*6"); // res는 30
    document.write("eval(₩"2*3+4*6₩")는 " + res + "<br>");
    let m = parseInt("32");
    document.write("parseInt(₩"32₩")는 " + m + "<br>");
    let n = parseInt("0x32");
    document.write("parseInt(₩"0x32₩")는 " + n + "<br><br>");

    // "hello"는 정수로 변환할 수 없으므로 parseInt("hello")는 NaN 리턴
    n = parseInt("hello");
    if(isNaN(n)) // true
        document.write("hello는 숫자가 아닙니다.");
}
</script>
</head>
<body>
<h3>eval(), parseInt(), isNaN()</h3>
<hr>
<script>
    evalParseIntIsNaN();
</script>
</body>
</html>
```



js_global_func2.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>함수 만들기</title>
<script>
function gugudan(n) { // 함수 작성
  let m = parseInt(n); // 문자열 n을 숫자로 바꿈
  if(isNaN(m) || m < 1 || m > 9) {
    alert("잘못입력하셨습니다.");
    return;
  }
  for(let i=1; i<=9; i++) { // i는 1~9까지 반복
    document.write(m + "x" + i + "=" + m*i + "<br>");
  }
}
</script>
</head>
<body>
<h3>구구단 출력 함수 만들기</h3>
<hr>
<script>
  let n = prompt("구구단 몇 단을 원하세요", ""); // n은 문자열
  gugudan(n); // 함수 호출
</script>
</body>
</html>
```

n이 1~9사이의 숫자가
아닌 경우 처리

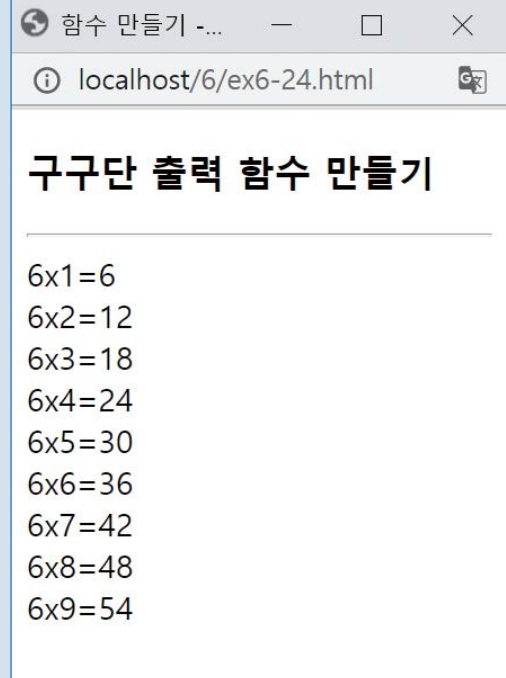
localhost 내용:

구구단 몇 단을 원하세요

시작

확인

취소



Mini Project: 숫자 맞추기 게임

- 프로그래밍이 가지고 있는 정수를 사용자가 알아맞히는 게임이다. 사용자가 답을 제시하면 프로그램은 자신이 저장한 정수와 비교하여 제시된 정수가 더 높은지 낮은지만을 알려준다. 정수의 범위를 1부터 100까지로 한정한다.




← → ↻ 127.0.0.1:55... ☆ 📁 📄 일시중지됨 ⋮

🔍 ⚙️ 설정 🔄 새 탭 📄 NAVER 🏠 전북대학교 >> 📁 모든 북마크

숫자 맞추기 게임

이 게임은 컴퓨터가 생성한 숫자를 맞추는 게임입니다. 숫자는 1부터 100 사이에 있습니다.



THE GUESSING 1-100 GAME

Guess the same number that the computer has guessed.
The number will range from 1 to 100.

숫자:

확인

추측횟수:

힌트:

js_guess.html

```
<html>
<head>
  <title>Number Guessing Game</title>
  <style>
    body {
      font-family: 'Arial', sans-serif;
      text-align: center;
      background-color: #f2f2f2;
      margin: 20px;
    }
    h2 {
      color: #333;
    }
    form {
      max-width: 400px;
      margin: 20px auto;
      background-color: #fff;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
  </style>
</head>
<body>
```

```
input[type="text"] {  
    width: 100%;  
    padding: 10px;  
    margin: 8px 0;  
    box-sizing: border-box;  
    border: 1px solid #ccc;  
    border-radius: 4px;  
}  
  
input[type="button"] {  
    width: 100%;  
    background-color: #4caf50;  
    color: white;  
    padding: 10px;  
    border: none;  
    border-radius: 4px;  
    cursor: pointer;  
}  
  
input[type="button"]:hover {  
    background-color: #45a049;  
}  
</style>
```

```
<script>
  let computerNumber = 53;
  let nGuesses = 0;

  function guess() {
    let result = "";
    let number = parseInt(document.getElementById("user").value);
    nGuesses++;

    if (number == computerNumber) result = "성공입니다.";
    else if (number < computerNumber) result = "낮습니다.";
    else result = "높습니다.";

    document.getElementById("result").value = result;
    document.getElementById("guesses").value = nGuesses;
    return true;
  }
</script>
</head>
```

```
<body>
  <h2>숫자 맞추기 게임</h2>
  <p>이 게임은 컴퓨터가 생성한 숫자를 맞추는 게임입니다.
    숫자는 1부터 100 사이에 있습니다.</p>
  <!-- 이미지 추가 -->
  
```

```
<form>
```

숫자:

```
<input type="text" id="user" size="5">
```

```
<input type="button" value="확인" onclick="guess();"><br>
```

추측횟수:

```
<input type="text" id="guesses" size="5"><br>
```

힌트:

```
<input type="text" id="result" size="16"><br>
```

```
</form>
```

```
</body>
```

```
</html>
```