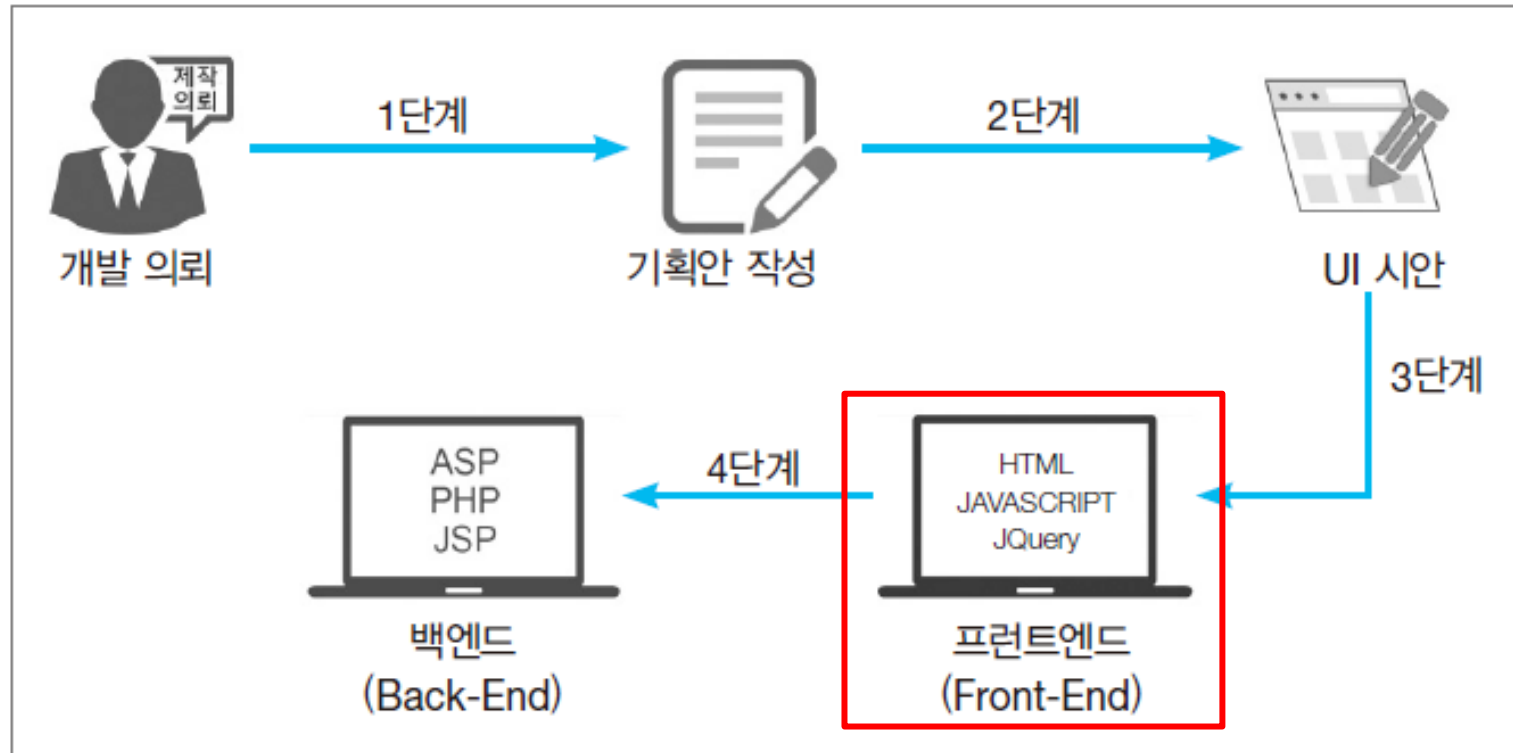


# 1.자바스크립트 시작하기



# 프론트엔드 개발의 이해



# 자바스크립트 언어



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

# 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

## 1. HTML 태그의 이벤트 리스너에 자바스크립트 코드 작성

onclick 이벤트 리스너 속성      자바스크립트 코드 (이미지를 banana.png로 교체)

```

```

# HTML 태그의 이벤트 리스너 속성에 자바스크립트 코드 작성

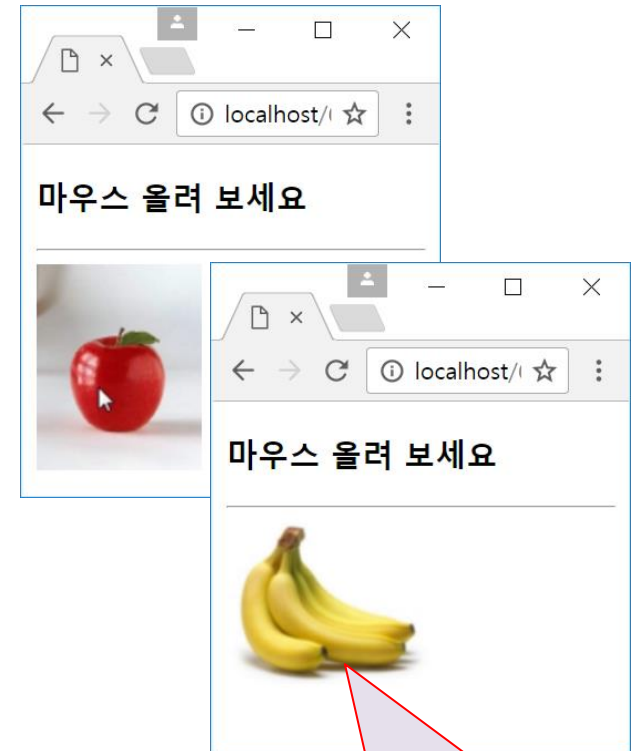
```
<!DOCTYPE html>
<html>
<head>
<title>이벤트 리스너 속성에 자바스크립트 코드</title>
</head>
<body>
<h3>마우스 올려 보세요</h3>
<hr>

</body>
</html>
```

이벤트 리스너  
속성

this는 현재 img 태그를  
가리키는 자바스크립트 키워드

자바스크립트  
코드



이미지에 마우스를 올리면 바나나로  
내리면 다시 사과로 바뀐다.

# <script> </script> 태그에 자바스크립트 작성



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

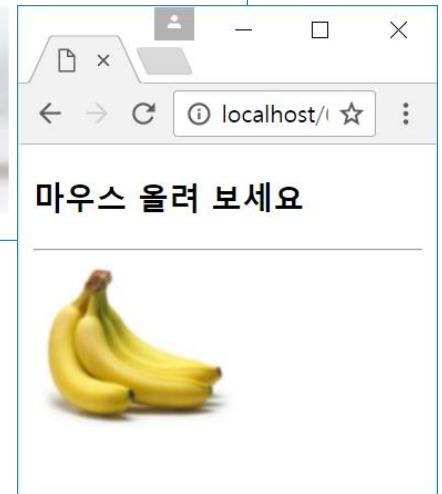
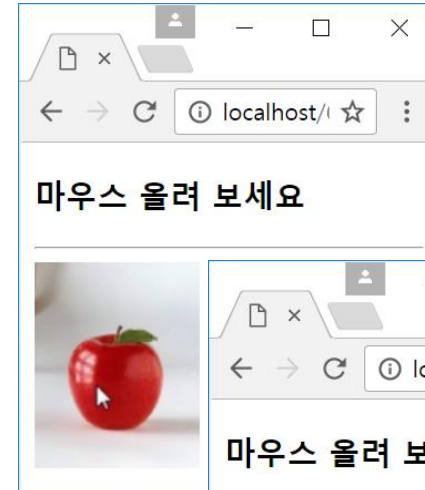
# <script> 태그에 자바스크립트 코드 작성

```
<!DOCTYPE html>
<html>
<head> <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 자바스크립트 키워드





# 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>
```

# 자바스크립트 파일 작성 및 불러오기

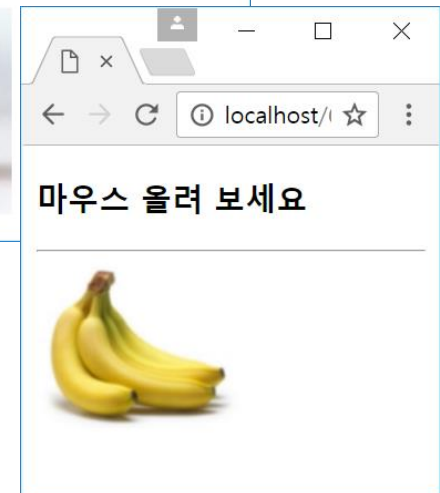
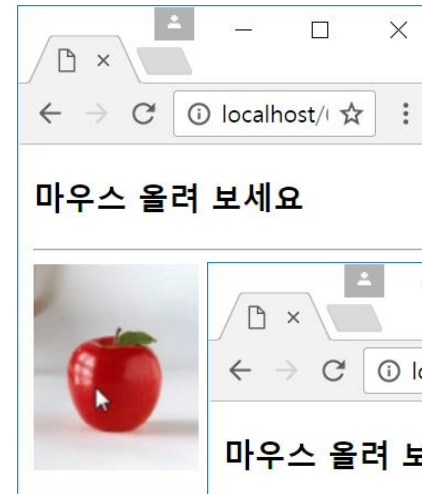


```
/* 자바스크립트 파일 lib.js */  
function over(obj) {  
    obj.src="media/banana.png";  
}  
function out(obj) {  
    obj.src="media/apple.png";  
}
```

lib.js

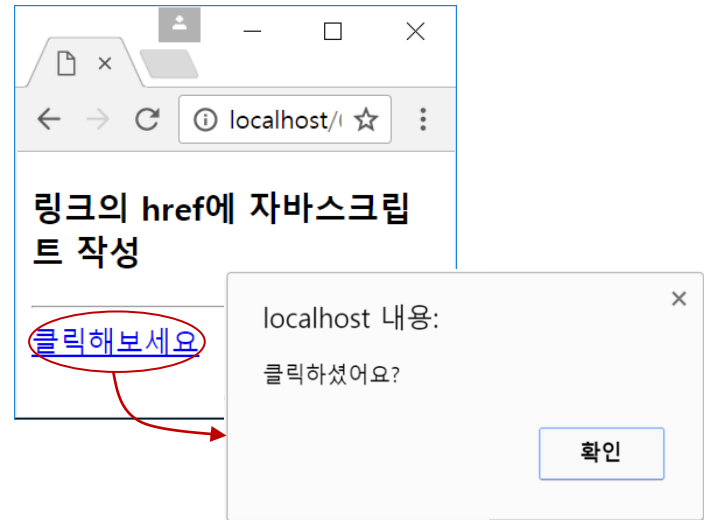
lib.js  
불러오기

```
<!DOCTYPE html>  
<html>  
<head> <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

```
<!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>
```



## 2.자바스크립트 기초 문법



# 자바스크립트 선언문



기본형

```
<script>  
  자바스크립트 코드;  
</script>
```

스크립트 영역

# 자바스크립트 주석 처리



기본형

```
//한 줄 설명글인 경우  
/*  
    설명글이 여러 줄인 경우  
    이렇게 처리합니다.  
*/
```

```
<!-- HTML 소스의 설명글은 이렇게 처리합니  
다 -->
```

# 내부 스크립트 외부로 분리하기



```
01: <!DOCTYPE html>
02: <html lang="ko">
03: <head>
04:   <meta charset="UTF-8">
05:   <title> 외부 자바스크립트 연동 </title>
06:   <script src="js/example.js"></script>
07: </head>
08: <body>
09: </body>
10: </html>
```

코딩해 보세요!

```
01: document.write("환영합니다");
```

• 완성 파일 js/example.js

# 코드 입력 시 주의할 점(1)



1. 자바스크립트는 대 · 소문자를 구분하여 작성합니다.

날짜 객체 생성: `New date( );` (X)

날짜 객체 생성: `new Date( );` (O)

2. 코드 한 줄을 작성한 후에는 세미콜론(;)을 쓰는 것이 좋습니다. 세미콜론을 쓰지 않으면 다음 예제처럼 한 줄에 2개의 코드를 작성할 경우 오류가 발생합니다.

`document.write("hi") document.write("bye")` (X)

`document.write("hi"); document.write("bye");` (O)



# 코드 입력 시 주의할 점(2)



3. 하지만 코드를 작성할 때는 한 줄에 한 문장만 작성하는 것이 가독성을 위해 좋습니다.
4. 문자형 데이터를 작성할 때는 큰따옴표(" ")와 작은따옴표(' ')의 겹침 오류를 주의해야 합니다.

## 큰따옴표 겹침 오류

```
document.write("책에 "자바스크립트는 대소문자를 구분해야 합니다"라고 나와 있다.");
```

## 잘된 예

```
document.write('책에 "자바스크립트는 대소문자를 구분해야 합니다"라고 나와 있다.');
```

```
document.write("책에 \"자바스크립트는 대소문자를 구분해야 합니다\"라고 나와 있다.");
```

5. 코드를 작성할 때 중괄호{} 또는 소괄호()의 짝이 맞아야 합니다.

```
document.write("welcome!");(X)
```

```
document.write("welcome!");(O)
```

# 변수



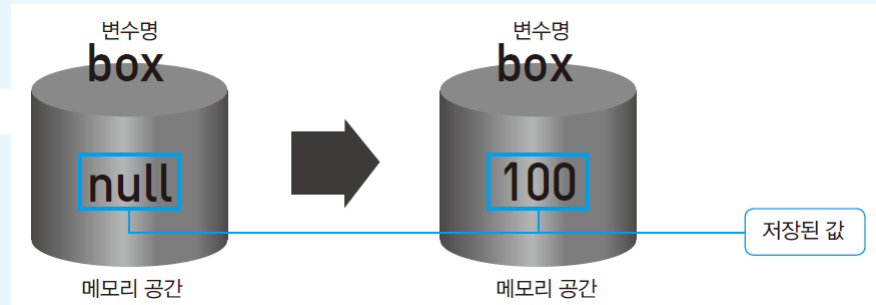
# 변수 선언



기본형

| var 변수명; 또는 var 변수명=값;

**var box;** ← 변수 선언  
box=100;



변수는 데이터를 담을 수 있는 **그릇**입니다.

# 변수에 저장할 수 있는 데이터 타입



## ① 문자형 데이터(String Type Data)

ex) var str = 'hello';

## ② 숫자형 데이터(Number Type Data)

ex) var num = 100;

## ③ 논리형 데이터(Boolean Type Data)

ex) var bool = true or false;

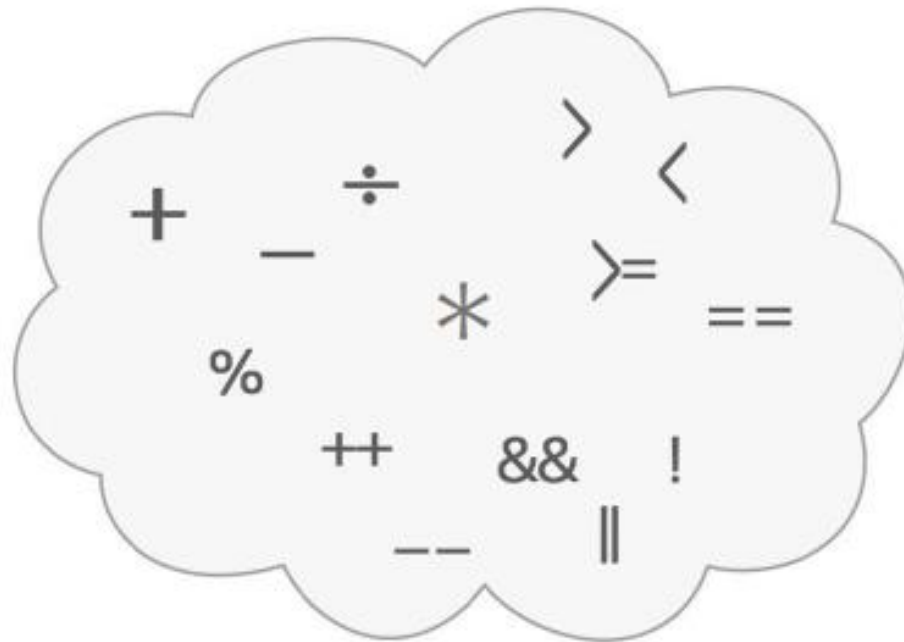
## ④ undefined

ex) var num; //undefined

## ⑤ null

ex) var num = null;

# 연산자란?



자바스크립트에서 사용하는 연산자의 종류

빼기, 더하기, 곱하기, 나누기, 비교 등...

# 산술 연산자

산술 연산자의 종류와 기본형

종류	기본형	설명
+	$A+B$	더하기
-	$A-B$	빼기
*	$A*B$	곱하기
/	$A/B$	나누기
%	$A\%B$	나머지

# 문자 연산자



## 기본형

문자형 데이터 + 문자형 데이터 = 하나의 문자형 데이터

ex) "do it " + "javascript" = "do it javascript";

문자형 데이터 + 숫자형 데이터 = 하나의 문자형 데이터

ex) "100" + 200 = "100200";

# 대입 연산자



## 대입 연산자의 종류

종 류	풀 이
$A = B$	$A = B$
$A += B$	$A = A + B$
$A *= B$	$A = A * B$
$A /= B$	$A = A / B$
$A \% = B$	$A = A \% B$



# 증감 연산자



기본형

① 변수의 값을 1만큼 감소시킵니다.

변수--; 또는 --변수;

② 변수의 값을 1만큼 증가시킵니다.

변수++; 또는 ++변수;

① 먼저 ㉠(B의 값을 1만큼 증가)가 실행되고, ㉡(증가된 B의 값을 A에 대입)가 실행됩니다.

var ㉠ A = ㉡ ++B

② 먼저 ㉠(B의 값을 A에 대입)가 실행되고, ㉡(B의 값을 1만큼 증가)가 실행됩니다.

var ㉠ A = ㉡ B++;

# 논리 연산자



## 논리 연산자의 종류


종류	설명
	or 연산자라 부르며, 피연산자 중 값이 하나라도 true가 존재하면 true로 결과값을 반환합니다.
&&	and 연산자라 부르며, 피연산자 중 값이 하나라도 false가 존재하면 false로 결과값을 반환합니다.
!	not 연산자라 부르며, 단항 연산자입니다. 피연산자의 값이 true이면 반대로 false로 결과값을 반환합니다.

# 삼항 조건 연산자



기본형 | 조건식 ? 자바스크립트 코드 1 : 자바스크립트 코드 2;

```
06: <script>
07:   var a = 10;
08:   var b = 3;
09:
10:   var result = a > b ? "javascript" : "hello";
11:   document.write(result); //javascript
12: </script>
```



The diagram illustrates the evaluation of the ternary operator expression `a > b ? "javascript" : "hello"`. A box labeled `true` has an arrow pointing to the `"javascript"` string, indicating that the condition `a > b` is true and the first branch of the operator is selected.

# 자바스크립트 입 출력



- 사용자 값 입력 : `prompt( )` 함수
- 알림 창 출력 : `alert( )` 함수
- 웹브라우저 화면 출력 : `document.write( )` 함수

# 3. 제어문



# 제어문이란?



조건문 (if 문 / else 문 / else if 문)

조건에 따라  
특정 코드를 실행시킬 수 있  
습니다.

선택문 (switch 문)

일치하는 경우의 값이  
있을 경우에만 특정 코드를  
실행시킬 수 있습니다.

## 제어문이란?

반복문 (while 문 / for 문)

코드를 지정한 횟수  
만큼 반복해서 실행시킬  
수 있습니다.

# 조건문



# if문



기본형

```
if(조건식){  
    자바스크립트 코드;  
}
```

## 적용 예제 1

```
var num=10;  
if(num<500){ //true를 반환합니다.  
    document.write("hello");  
}
```



# else문



기본형

```
if(조건식){
    자바스크립트 코드1;
}else{
    자바스크립트 코드2;
}
```

```
06: <script>
07:   var num = prompt("당신이 좋아하는 숫자는?", "0");
08:
09:   if(num % 2 == 0) {    //짝수일 경우에 실행
10:     document.write("당신이 좋아하는 숫자는 짝수입니다.");
11:   } else {              //홀수일 경우에 실행
12:     document.write("당신이 좋아하는 숫자는 홀수입니다.");
13:   }
14: </script>
```

# else if문



기본형

```
if(조건식1){  
    코드1;  
}else if(조건식2){  
    코드2;  
}else if(조건식3){  
    코드3;  
}else if(조건식4){  
    코드4;  
}else if(조건식5){  
    코드5;  
}else{  
    코드6;  
}
```

# 중첩 if문



기본형

```
if(조건식1){  
    if(조건식2){  
        자바스크립트 코드;  
    }  
}
```

```
13:  if(id == user_id) { — 아이디가 일치하면 실행됩니다.  
14:      if(pw == user_pw) {  
15:          document.write(user_id+"님 반갑습니다!"); — 비밀번호가 일치하면  
16:      } else { — 실행됩니다.  
17:          alert("비밀번호가 일치하지 않습니다."); — 비밀번호가 일치하지  
18:          location.reload( ); — 브라우저 새로 고침 않으면 실행됩니다.  
19:      }  
20:  } else {  
21:      alert("아이디가 일치하지 않습니다."); — 아이디가 일치하지 않  
22:      location.reload( ); — 으면 실행됩니다.  
23:  }
```

# 선택문



# quiz



- 입력창에 아이디 와 비밀번호를 입력 받고 모두 일치하면 다음과 같이 출력하는 스크립트를 작성 하시오.

easy1004님 반갑습니다!

# switch문



기본형

```
var 변수=초깃값;
```

```
switch(변수){
```

```
case 값1: 코드1;
```

```
break;
```

```
case 값2: 코드2;
```

```
break;
```

```
case 값3: 코드3;
```

```
break;
```

```
case 값4: 코드4;
```

```
break;
```

```
default: 코드5;
```

```
}
```

← switch문을 만나면

← case를 하나씩 검사

← 변수가 모든 case의 값과 일치하지 않으면

# 반복문



# while문



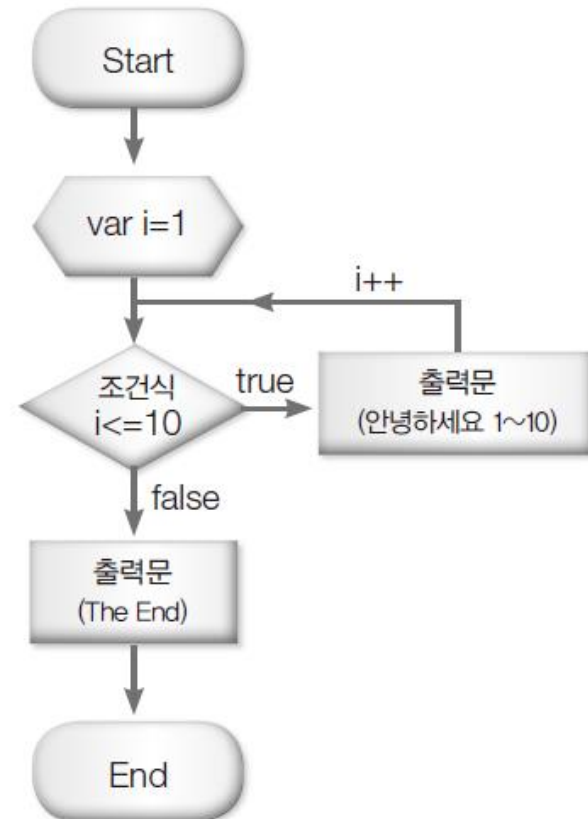
기본형

```
var 변수=초깃값;  
while(①③조건식){
```

②

```
    자바스크립트 코드;  
    증감식;
```

```
}
```





# quiz



- 20~10까지의 숫자 중 짝수는  
파란색, 홀수는 빨간색으로 표시.

20

19

18

17

16

15

14

13

12

11

10

# do while문



기본형

```
var 변수=초깃값;
```

```
do{
```

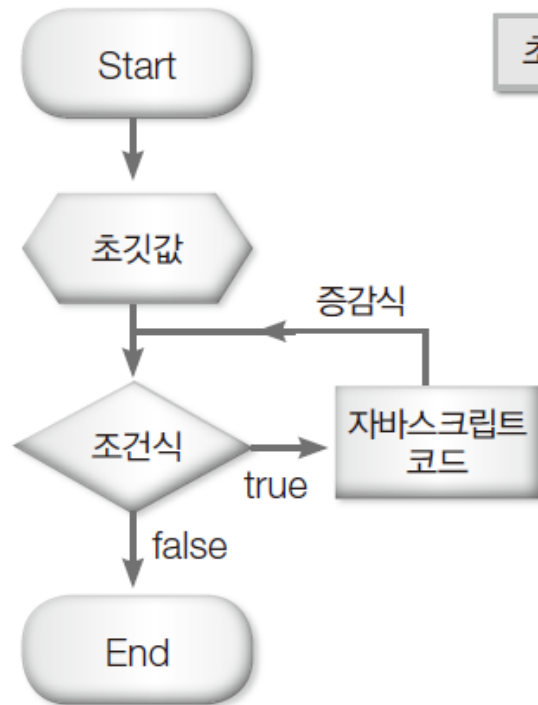
```
    자바스크립트 코드;
```

```
    증감식;
```

```
}while(조건식)
```

```
06: <script>
07:   var i = 10;
08:   do {
09:     document.write("hello!!");
10:   } while ( i < 3 )
11: </script>
```

# for문



초깃값 → 조건식 → 자바스크립트 코드 → 증감식 → 조건식

기본형

```
for(초깃값; 조건식; 증감식){  
    자바스크립트 코드;  
}
```

# break문



## 기본형

```
for(초깃값; 조건식; 증감식){  
    break; //반복문을 강제로 종료합니다.  
    자바스크립트 코드;  
}
```

```
var 변수=초깃값;  
while(조건식){  
    break; //반복문을 강제로 종료합니다.  
    자바스크립트 코드;  
    증감식;  
}
```

# continue문



## 기본형

```
for(초깃값; 조건식; 증감식){  
    continue;  
    자바스크립트 코드;  
}
```

```
var 변수=초깃값;  
while(조건식){  
    증감식;  
    continue;  
    자바스크립트 코드;  
}
```

# quiz



- 반복문을 이용한 구구단 출력

$$5 \times 1 = 5$$

$$5 \times 2 = 10$$

$$5 \times 3 = 15$$

$$5 \times 4 = 20$$

$$5 \times 5 = 25$$

$$5 \times 6 = 30$$

$$5 \times 7 = 35$$

$$5 \times 8 = 40$$

$$5 \times 9 = 45$$

# quiz



- 중첩 for 문을 이용하여 다음과 같은 표를 완성 하시오.

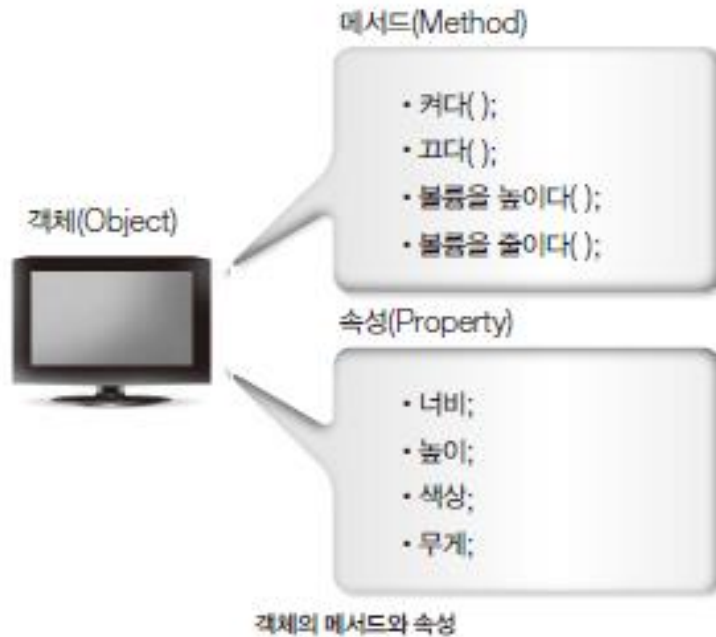
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

# 4. 객체





# 객체란?



객체 = 기능과 속성을 가지는 것

# 객체의 종류



## ① 내장 객체

문자(String), 날짜(Date), 배열(Array), 수학(Math), ...

## ② 브라우저 객체 모델(BOM)

- window, screen, location, history, navigator, ...
- document와 location 객체의 상위 객체

## ③ 문서 객체 모델(DOM)

- <html>, <head>, <body>, ...
- HTML 문서 구조

# 내장 객체



# 내장 객체 생성하기



참조 변수(인스턴스 이름) = new 생성 함수( )

# 수학 객체 — 메서드, 속성



종류	설명
Math.abs(숫자)	숫자의 절댓값을 반환합니다.
Math.max(숫자 1, 숫자 2, 숫자 3, 숫자 4)	숫자 중 가장 큰 값을 반환합니다.
Math.min(숫자 1, 숫자 2, 숫자 3, 숫자 4)	숫자 중 가장 작은 값을 반환합니다.
Math.pow(숫자, 제곱값)	숫자의 거듭제곱값을 반환합니다.
Math.random( )	0~1 사이의 난수를 반환합니다.
Math.round(숫자)	소수점 첫째 자리에서 반올림하여 정수를 반환합니다.
Math.ceil(숫자)	소수점 첫째 자리에서 무조건 올림하여 정수를 반환합니다.
Math.floor(숫자)	소수점 첫째 자리에서 무조건 내림하여 정수를 반환합니다.
Math.sqrt(숫자)	숫자의 제곱근값을 반환합니다.
Math.PI	원주율 상수를 반환합니다.

# quiz



## ■ 가위 바위 보 게임 만들기

컴퓨터 가위, 바위, 보 맞추기



가위 ( 2 )

틀렸네요!

다음기회 또 도전하세요!!

# 배열 객체(1)



- 여러 개의 데이터를 하나에 장소에 저장 하기

- ❶ `var 참조 변수=new Array( );`
- ❷ `var 참조 변수=new Array(값1, 값2, 값3, ...값n);`
- ❸ `var 참조 변수=[값1, 값2, 값3, ...값n];`

# 배열 객체(2) — 메서드, 속성



종류	설명
join(연결 문자)	배열 객체의 데이터를 연결 문자 기준으로 1개의 문자형 데이터로 반환합니다.
reverse( )	배열 객체의 데이터 순서를 거꾸로 바꾼 후 반환합니다.
sort( )	배열 객체의 데이터를 오름차순으로 정렬합니다.
slice(index1, index2)	배열 객체의 데이터 중 원하는 인덱스 구간만큼 잘라서 배열 객체로 가져옵니다.
splice( )	배열 객체의 지정 데이터를 삭제하고 그 구간에 새 데이터를 삽입할 수 있습니다.
concat( )	2개의 배열 객체를 하나로 결합합니다.
pop( )	배열에 저장된 데이터 중 마지막 인덱스에 저장된 데이터를 삭제합니다.
push(new data)	배열 객체의 마지막 인덱스에 새 데이터를 삽입합니다.
shift( )	배열 객체에 저장된 데이터 중 첫 번째 인덱스에 저장된 데이터를 삭제합니다.
unshift(new data)	배열 객체의 가장 앞의 인덱스에 새 데이터를 삽입합니다.
length	배열에 저장된 총 데이터의 개수를 반환합니다.



# 문자열 객체



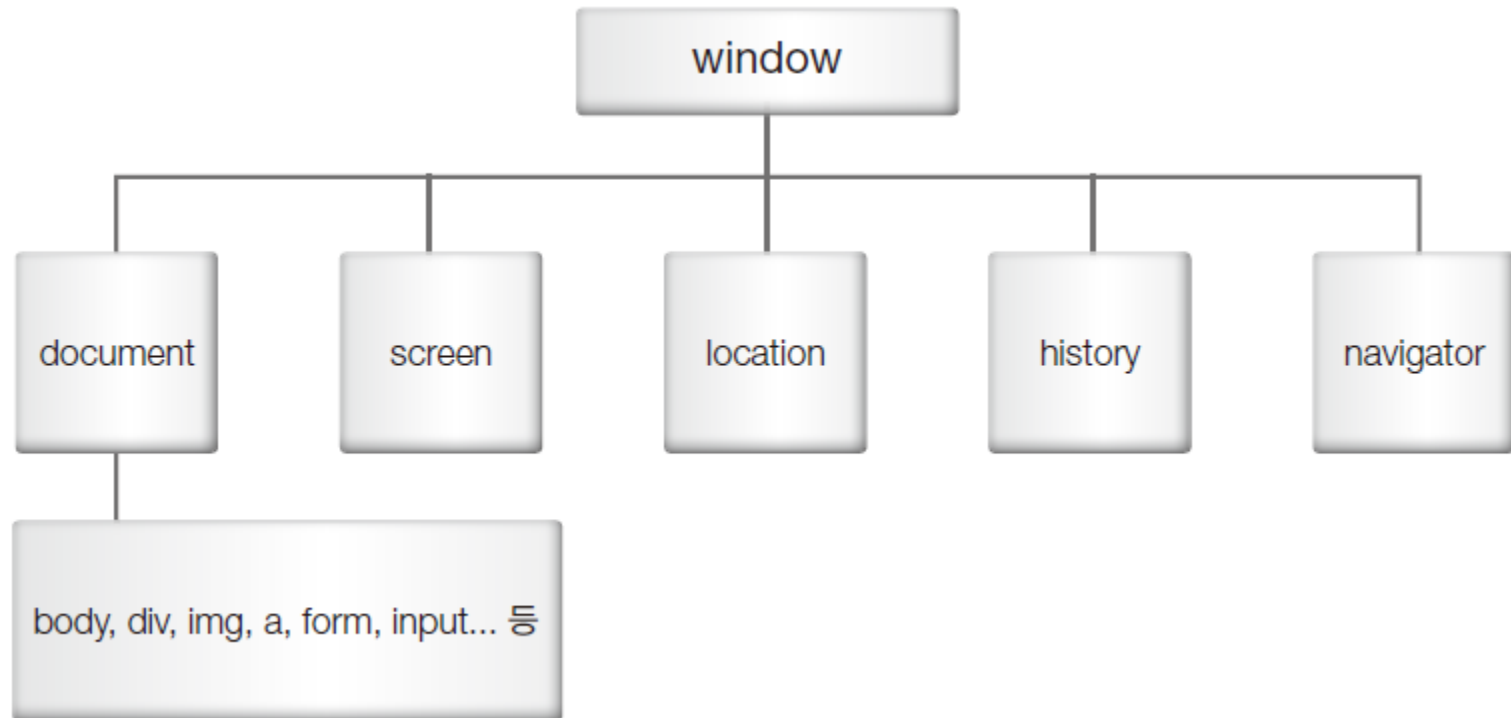
```
var 참조 변수=new String(문자형 데이터)
```

# 브라우저 객체 모델



- 브라우저에 내장된 객체
- window는 브라우저 객체의 최상위 객체

# 브라우저 객체



# 브라우저 객체 — 메서드



종류	설명
<code>open("URL", "새 창 이름", "새 창 옵션")</code>	URL 페이지를 새 창으로 나타냅니다.
<code>alert(data)</code>	경고 창을 나타내고 데이터를 보여줍니다. 방문자가 [확인] 버튼을 누르면 <code>alert()</code> 를 사용한 다음 위치의 코드를 수행합니다.
<code>prompt("질문", "답변")</code>	질문과 답변으로 질의응답 창을 나타냅니다.
<code>confirm("질문 내용")</code>	질문 내용으로 확인이나 취소 창을 나타냅니다. [확인] 버튼을 누르면 <code>true</code> 를 반환하고, [취소] 버튼을 누르면 <code>false</code> 를 반환합니다.
<code>moveTo(x, y)</code>	지정한 새 창의 위치를 이동합니다.
<code>resizeTo(width, height)</code>	지정한 새 창의 크기를 변경합니다.
<code>setInterval(function( ) { 자바스크립트 코드 }, 일정 시간 간격)</code>	지속적으로 일정한 시간 간격으로 함수를 호출하여 코드를 실행합니다.
<code>setTimeout(function( ) { 자바스크립트 코드 }, 일정 시간 간격)</code>	단 한 번 일정한 시간 간격으로 함수를 호출하여 코드를 실행합니다.

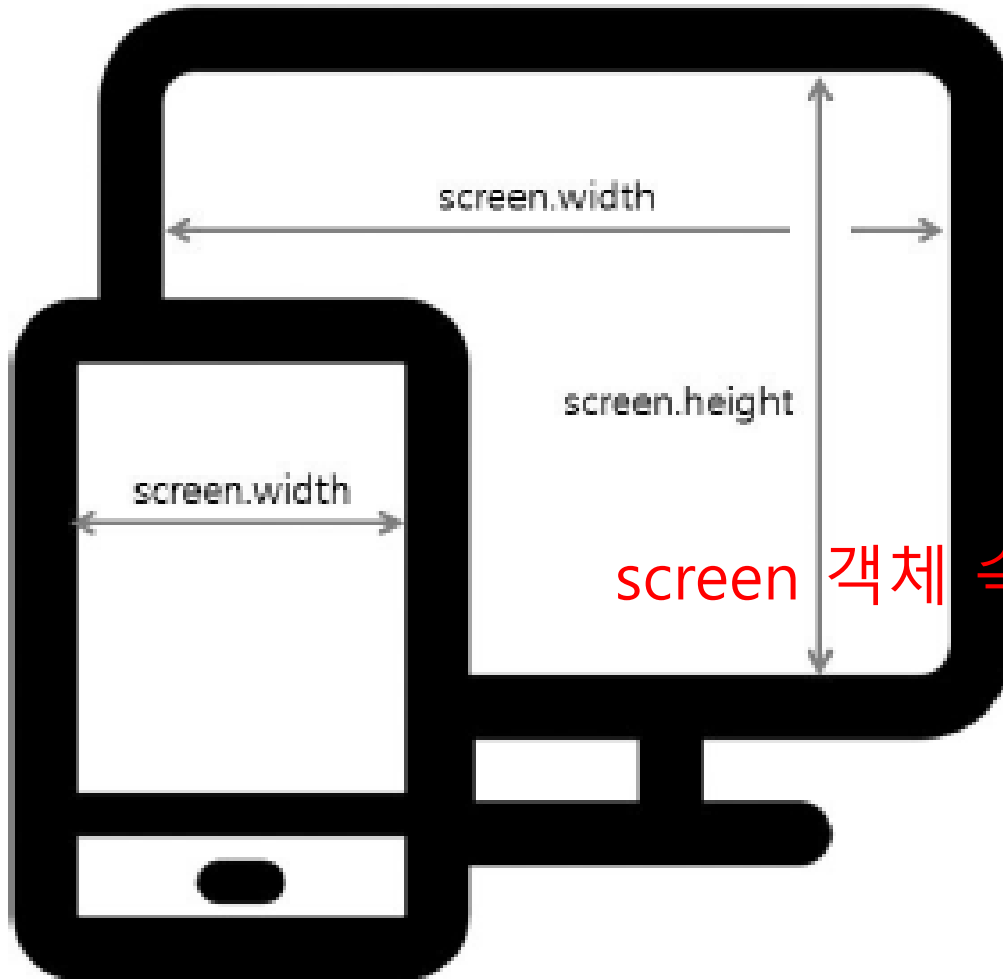
# 새 창의 옵션



! 팝업 반드시 해제하고 진행

속성	설명
① width	새 창의 너비를 설정합니다.
② height	새 창의 높이를 설정합니다.
③ left	새 창의 수평(X축) 위치를 설정합니다.
④ top	새 창의 수직(Y축) 위치를 설정합니다.
⑤ scrollbars	새 창의 스크롤바의 숨김/노출을 설정합니다(숨김 = no, 노출 = yes).
location	새 창의 URL 주소 입력 영역의 숨김/노출을 설정합니다(숨김 = no, 노출 = yes).
status	새 창의 상태 표시줄 영역의 숨김/노출을 설정합니다(숨김 = no, 노출 = yes).
toolbars	새 창의 도구 상자 영역의 숨김/노출을 설정합니다(숨김 = no, 노출 = yes).

# screen 객체



screen 객체 속성 = width, height, ...

# screen 객체 — 속성



종류	설명
screen.width	화면의 너비값을 반환합니다.
screen.height	화면의 높이값을 반환합니다.
screen.availWidth	작업 표시줄을 제외한 화면의 너비값을 반환합니다.
screen.availHeight	작업 표시줄을 제외한 화면의 높이값을 반환합니다.
screen.colorDepth	사용자 모니터가 표현 가능한 컬러 bit를 반환합니다.

# location 객체 — 속성



종류	설명
location.href	주소 영역의 참조 주소를 설정하거나 URL을 반환합니다. 예) <code>http://www.easyspub.co.kr:80/Main/pub#view</code>
location.hash	URL의 해시값(#에 명시된 값)을 반환합니다. 예) <code>http://www.easyspub.co.kr/Main/pub#view</code>
location.hostname	URL의 호스트 이름을 설정하거나 반환합니다. 예) <code>http://www.easyspub.co.kr:80/</code>
location.host	URL의 호스트 이름과 포트 번호를 반환합니다. 예) <code>http://www.easyspub.co.kr:80/</code>
location.protocol	URL의 프로토콜을 반환합니다. 예) <code>http://www.easyspub.co.kr:80/</code>
location.search	URL의 쿼리(요청값)를 반환합니다. 예) <code>http://www.easyspub.com?pageNum=1&amp;sort=DESC</code>
location.reload()	마치 브라우저에서 <code>[F5]</code> 키를 누른 것처럼 새로 고침합니다.



# 5.함수



# 함수란?



## 변수

- 1개의 데이터만 저장합니다.
- var라는 키워드를 이용하여 선언합니다.
- 문자형, 숫자형, 논리형 데이터를 보관합니다.
- 객체를 참조합니다.

VS

## 함수

- 자바스크립트 코드를 저장합니다.
- function이라는 키워드를 이용하여 선언합니다.
- 출력문, 제어문 등의 코드를 저장하고 데이터를 반환합니다.



변수(var)



함수(function)

# 기본 함수 정의문 — 정의, 사용



```
function myFnc( ) {  
    document.write( "hello~", "<br>" );  
    document.write( "welcome", "<br>" );  
}
```

myFnc( );

myFnc( );

함수가 2회 호출되어 코드  
내용을 2회 실행합니다.

# 일반 함수 정의 vs 익명 함수 선언 참조



## 전문가의 조언

### 일반 함수 정의 방식과 익명 함수 선언 참조 방식의 차이점

일반 함수 정의는 함수 호출 시 호이스팅(hoisting) 기술을 지원합니다. 그러나 익명 함수 선언 참조 방식은 호이스팅을 지원하지 않습니다. 호이스팅을 적용하면 함수 정의문보다 호출문이 먼저 나와도 함수 정의문을 끌어올려 함수를 호출합니다.

A. 일반 함수 정의 방식 (정상 작동)	B. 익명 함수 선언 참조 방식 (오류 발생)
<div>Ⓐ testFnc(); ← 호출문</div> <div>↓ 호이스팅(hoisting)</div> <div>Ⓑ function testFnc() {     Ⓒ 자바스크립트 코드; } ← 함수 정의문</div>	<div>Ⓓ testFnc();</div> <div>Ⓔ var testFnc = function () {     Ⓕ 자바스크립트 코드; }</div>

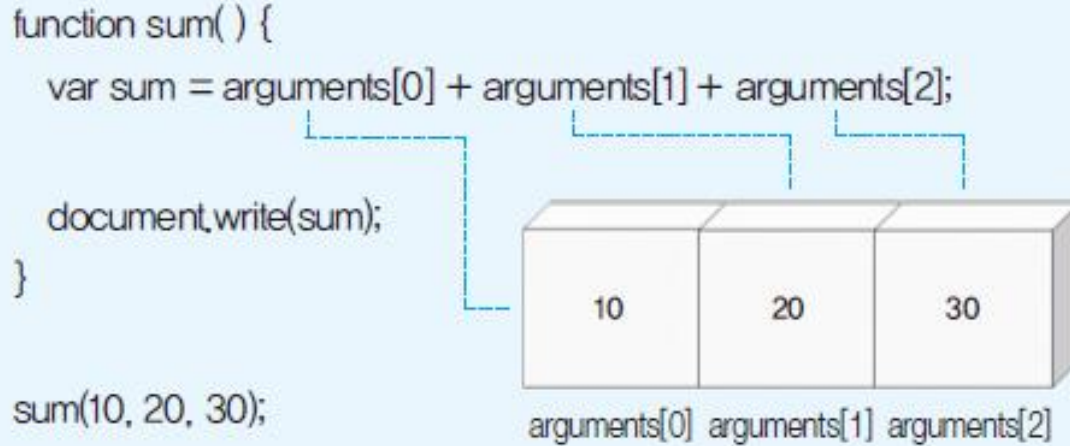
# 매개변수가 있는 함수 정의문



기본형

```
function 함수명(①매개변수 1, ②매개변수 2,...③매개변수 n)  
    자바스크립트 코드;  
}  
함수명(a데이터 1, b데이터 2,...c데이터 n);
```

# 매개변수 없이 함수에 전달된 값 받아오기



# return



# 데이터를 반환하고 강제 종료하는 return 문



기본형

```
function 함수명() {
```

```
  자바스크립트 코드1; ②
```

```
  return 데이터(값); ③
```



```
  자바스크립트 코드2;
```

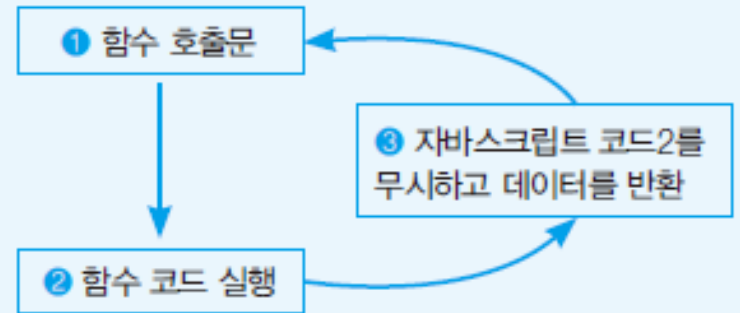
```
}
```

```
var 변수 = 함수명(); ①
```

① 함수 호출문

② 함수 코드 실행

③ 자바스크립트 코드2를  
무시하고 데이터를 반환





# 재귀 함수 호출



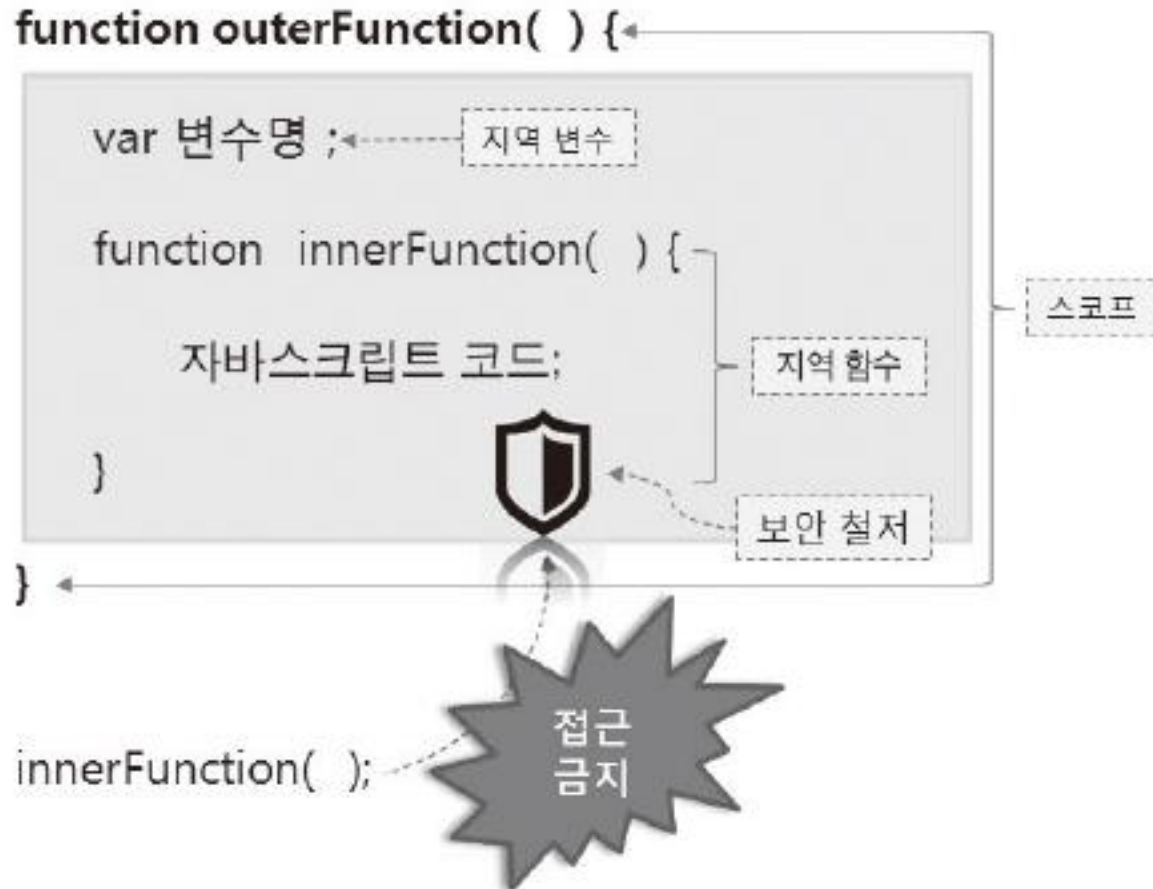
기본형

```
function myFnc( ){  
    자바스크립트 코드;  
    myFnc( );  
}  
myFnc( );
```

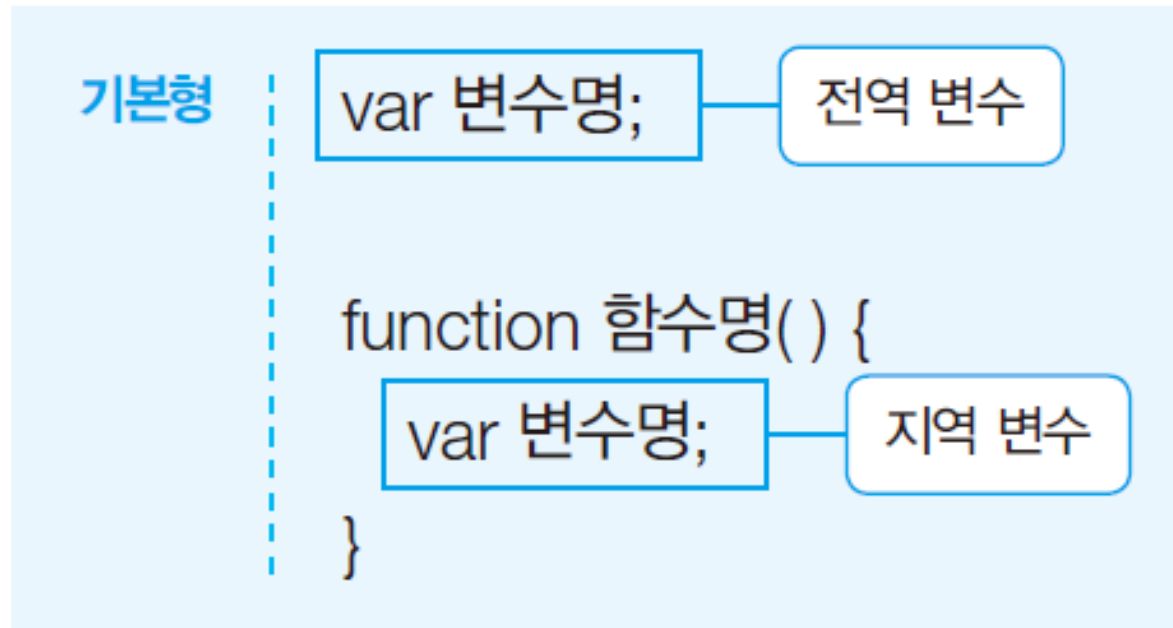
# 스코프(scope)



# 함수 스코프란?



# 전역 변수와 지역 변수의 개념과 차이



# 전역 함수와 지역 함수의 차이



기본형

```
function 함수명1() {  
  자바스크립트 코드;  
}
```

전역 함수

```
function 함수명2() {
```

```
  function 함수명3() {  
    자바스크립트 코드;  
  }
```

지역 함수

```
}
```

# 생성자 함수의 활용



# 객체 생성자 함수

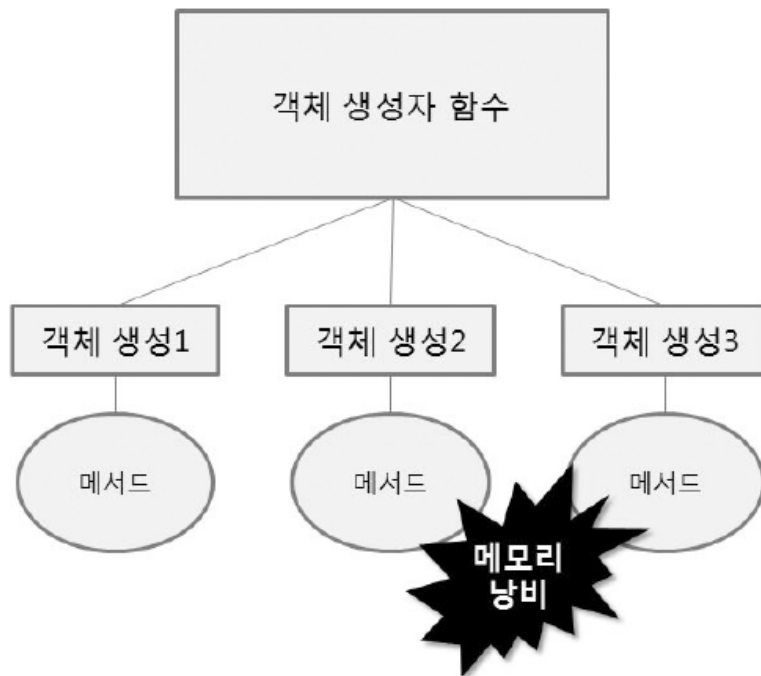


기본형

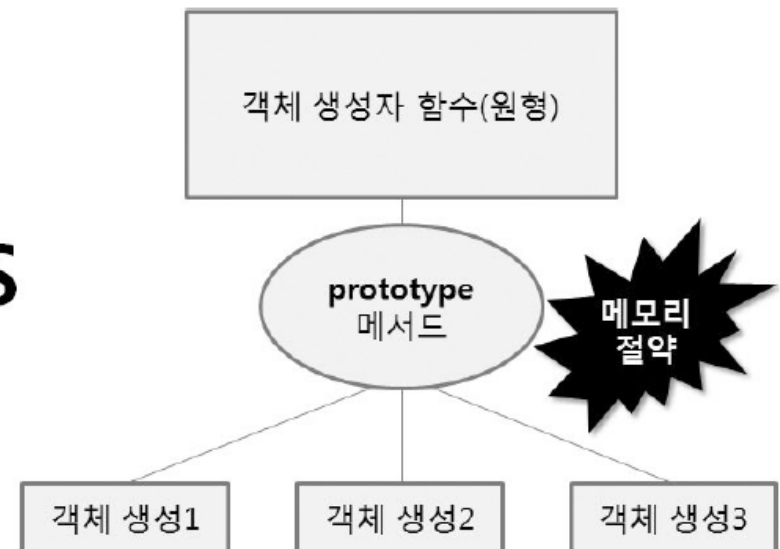
```
function 함수명(매개변수1, 매개변수2, ...매개변수n) {    //객체 생성자 함수
    this.속성명 = 새 값;
    this.함수명 = function() {
        자바스크립트 코드;
    }
}

var 참조 변수(인스턴스 네임) = new 함수명();    //객체 생성
    ↓
var 참조 변수 = { 속성 : 새 값, 함수명 : function() { ... } }
```

# 메모리 절약을 위한 프로토타입 사용하기(1)



VS





# 메모리 절약을 위한 프로토타입 사용하기(2)



기본형

```
function 함수명(매개변수1, 매개변수2, ...매개변수n) {  
    this.속성명 = 새 값;  
}
```

```
함수명.prototype.함수명 = function() {  
    자바스크립트 코드;  
}
```

```
var 참조 변수(인스턴스 네임) = new 함수명( );
```

# 자바스크립트 내장 함수



# 내장 함수



종류	설명	사용 예
<code>encodeURIComponent()</code>	문자를 유니 코드값으로 인코딩합니다. (영문, 숫자, 일부 기호(, / ? : @ & = + \$)는 제외)	<code>encodeURIComponent("?query=값");</code> → <code>"?query=%EA%B0%91"</code>
<code>encodeURIComponentComponent()</code>	문자를 유니 코드값으로 인코딩합니다(영문, 숫자 제외).	<code>encodeURIComponentComponent("?query=값")</code> → <code>"%3Fquery%3D%EA%B0%91"</code>
<code>decodeURI()</code>	유니 코드값을 디코딩해 다시 문자화합니다.	<code>decodeURI("?query=%EA%B0%91")</code> → <code>"?query=값"</code>
<code>decodeURIComponent()</code>	유니 코드값을 디코딩해 다시 문자화합니다.	<code>decodeURIComponent("%3Fquery%3D%EA%B0%91")</code> → <code>"?query=값"</code>
<code>parseInt()</code>	문자열 데이터를 정수형 데이터로 반환합니다.	<code>parseInt("5,12")</code> → 5 <code>parseInt("15px")</code> → 15
<code>parseFloat()</code>	문자열 데이터를 실수형 데이터로 반환합니다.	<code>parseFloat("5,12")</code> → 5.12 <code>parseFloat("65.5%")</code> → 65.5
<code>String()</code>	문자형 데이터로 반환합니다.	<code>String(5)</code> → "5"
<code>Number()</code>	숫자형 데이터로 반환합니다.	<code>Number("5")</code> → 5
<code>Boolean()</code>	논리형 데이터로 반환합니다.	<code>Boolean(5)</code> → true <code>Boolean(null)</code> → false
<code>isNaN()</code>	is Not a Number의 약자이며 숫자가 아닌 문자가 포함되어 있으면 true를 반환합니다.	<code>isNaN("5-3")</code> → true <code>isNaN("53")</code> → false
<code>eval()</code>	문자형 데이터를 따옴표가 없는 자바스크립트 코드로 처리합니다	<code>eval("15 + 5")</code> → 20