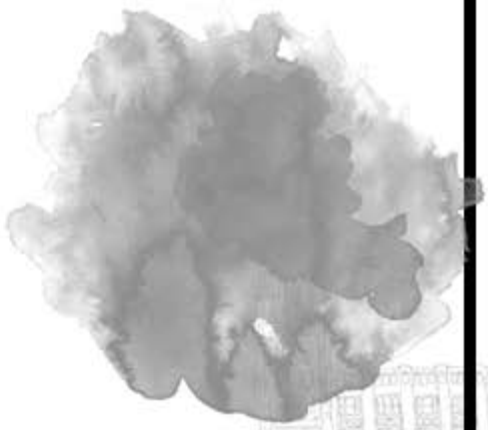
함수는 객체다

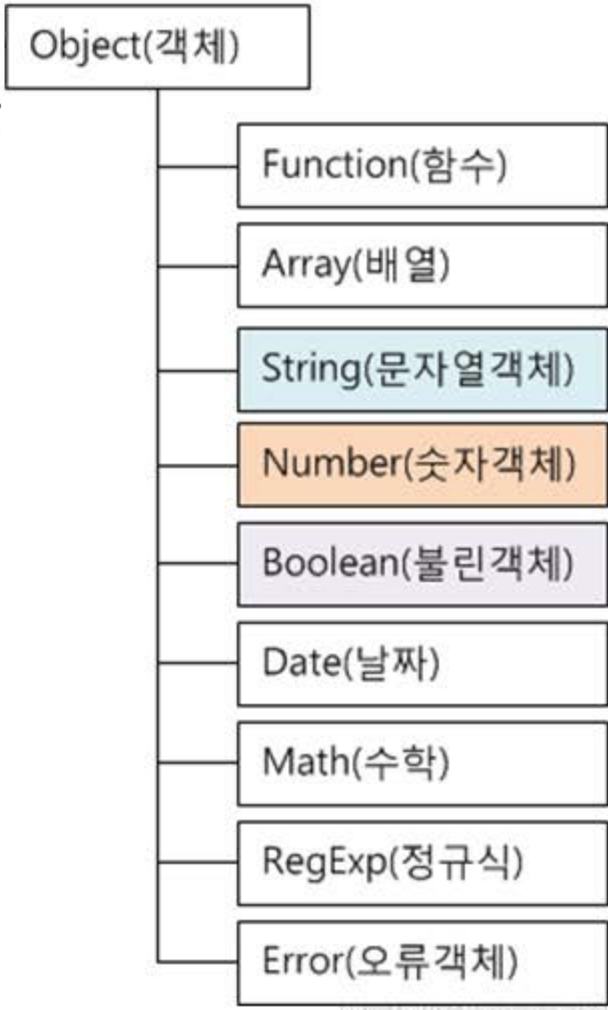
학습목표

- ✓ 함수란?
- ✓ 함수의 프로퍼티와 메서드
  - length 프로퍼티
  - callee 프로퍼티
  - prototype 프로퍼티
  - constructor 프로퍼티
- ✓ 함수 계층도
- ✓ 함수의 종류
  - 콜백 함수
  - 즉시 함수
  - 내부 함수
  - 클로저 함수
  - 재귀 함수
  - 생성자 함수



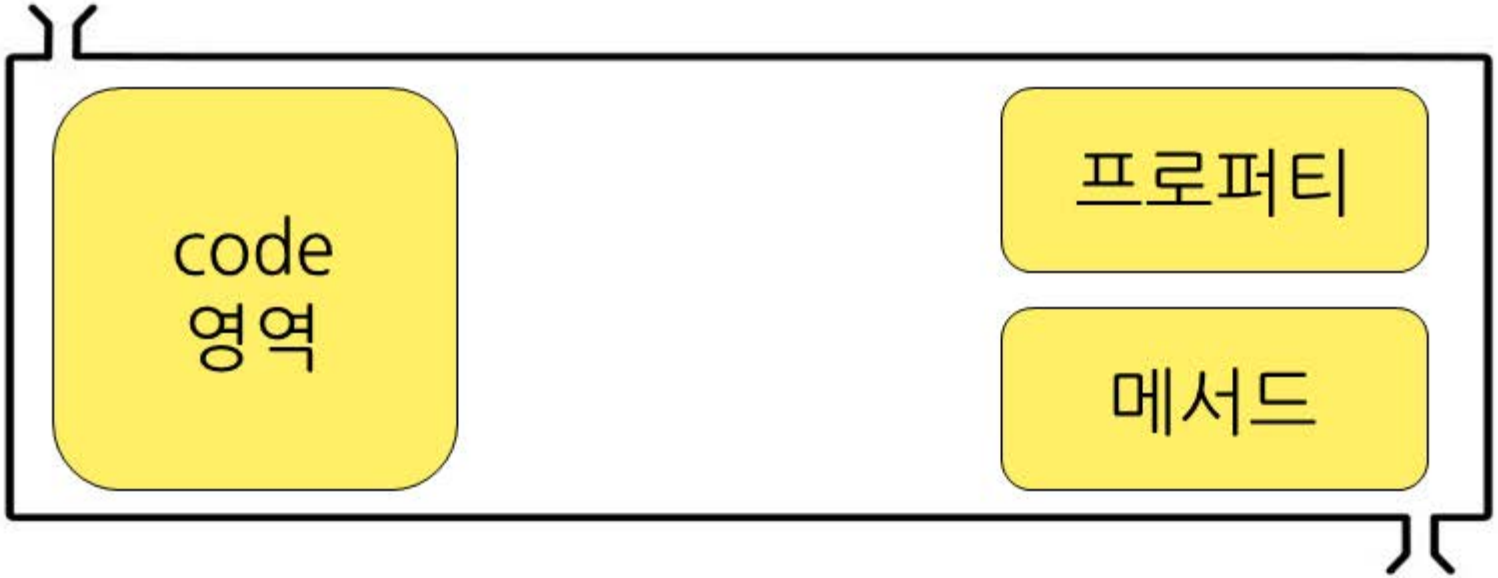
함수란?

- 함수는 박스다.
  - 함수는 입력한 값을 받아 처리한 결과를 반환한다
  - ex) alert( ), prompt( )
- 함수는 값이다.
  - 변수에 함수 대입 가능
  - 프로퍼티에 함수 사용 가능(메서드)
  - 배열 요소로 함수 사용 가능
  - **매개변수로 함수 사용 가능**
  - **리턴값으로 함수 사용 가능(클로저)**
- 함수는 객체다
  - 함수는 프로퍼티를 가질 수 있다.
  - 함수는 메서드를 가질 수 있다.
  - 코드영역을 갖는다.



함수는 객체다

- 함수 = 코드영역 + 프로퍼티 + 메서드

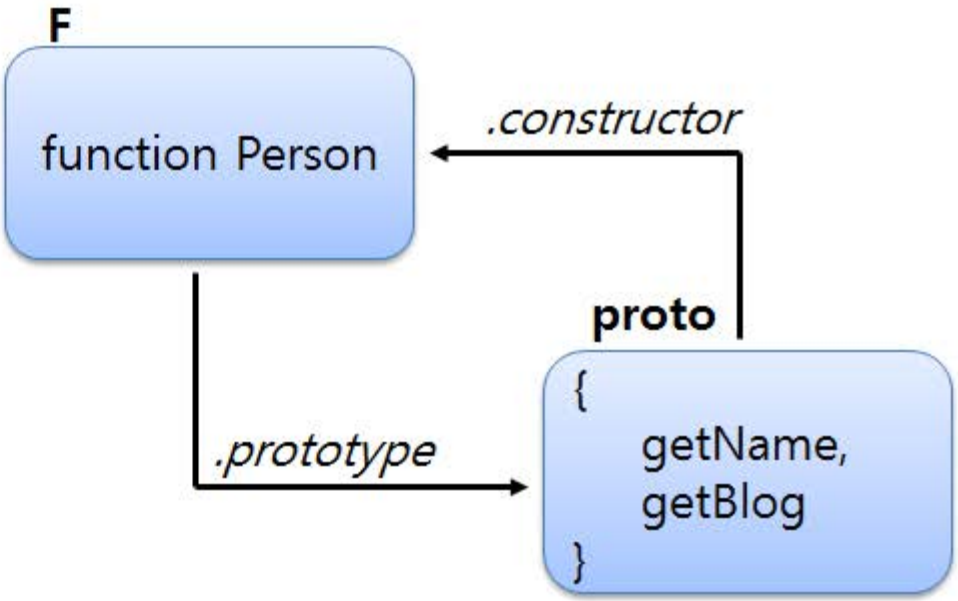


js10-01 함수는 객체다. 객체는 프로퍼티와 메서드를 갖는다.
그러므로 함수는 프로퍼티와 메서드를 갖는다.

```
<script>
    var plus = function (a, b) {
        return a + b;
    };

    plus.result = plus(1,3);
    plus.status = 'OK';
    plus.desc = 'I am function';

    console.log( plus(1,3) ) ;
    console.log( plus.result ) ;
    console.log( plus.status ) ;
    console.log( plus.desc ) ;

    plus.getDesc = function () {
        return this.desc;
    };
    console.log( plus.getDesc() ) ;
</script>
```

함수의 프로퍼티와 메서드

- 함수 기본 프로퍼티
  - length 프로퍼티
  - callee 프로퍼티
  - prototype 프로퍼티
  - constructor 프로퍼티



- 함수 기본 메서드
  - toString() 메서드
  - apply() 메서드
  - call() 메서드
  - bind() 메서드

```
<script type="text/javascript">
    var myFunction = function () {
        return true;
    }

    console.log( myFunction.prototype );
    console.log( myFunction.prototype.constructor );
</script>
```

함수의 프로퍼티: length

- 매개변수의 개수

```
function func0(){
}
function func1(x){
    return x;
}
function func2(x,y){
    return x+y;
}
function func3(x,y,z){
    return x+y+z;
}

console.log( 'func0.length - ' + func0.length );
console.log( 'func1.length - ' + func1.length );
console.log( 'func2.length - ' + func2.length );
console.log( 'func3.length - ' + func3.length );
```

예제. 함수의 매개변수 length

```
<script>
    function func0( ){
    }
    function func1(x){
        return x;
    }
    function func2(x,y){
        return x+y;
    }
    function func3(x,y,z){
        return x+y+z;
    }

    console.log( 'func0.length - ' + func0.length );
    console.log( 'func1.length - ' + func1.length );
    console.log( 'func2.length - ' + func2.length );
    console.log( 'func3.length - ' + func3.length );
</script>
```

함수의 프로퍼티: callee

- 현재 실행 중인 함수를 가르킨다
- 익명 함수에서 재귀 호출 구현할 때 유용

```
var makeFactorial = function () {
    return function(x) {
        if( x<= 1)
            return 1;
        else
            return x*arguments.callee(x-1);
    };
}

var result = makeFactorial();
console.log( result(5) );  // (출력값) 120
```

예제. 함수 매개변수 callee

```
<script>
    var makeFactorial = function () {
        return function(x) {
            if( x<= 1)
                return 1;
            else
                return x*arguments.callee(x-1);
        };
    }

    var result = makeFactorial();
    console.log( result(5) );  // (출력값) 120
</script>
```

함수는 값이다

- 변수에 함수 대입 가능
- 프로퍼티에 함수 대입 가능(메서드)
- 배열 요소로 함수 사용 가능

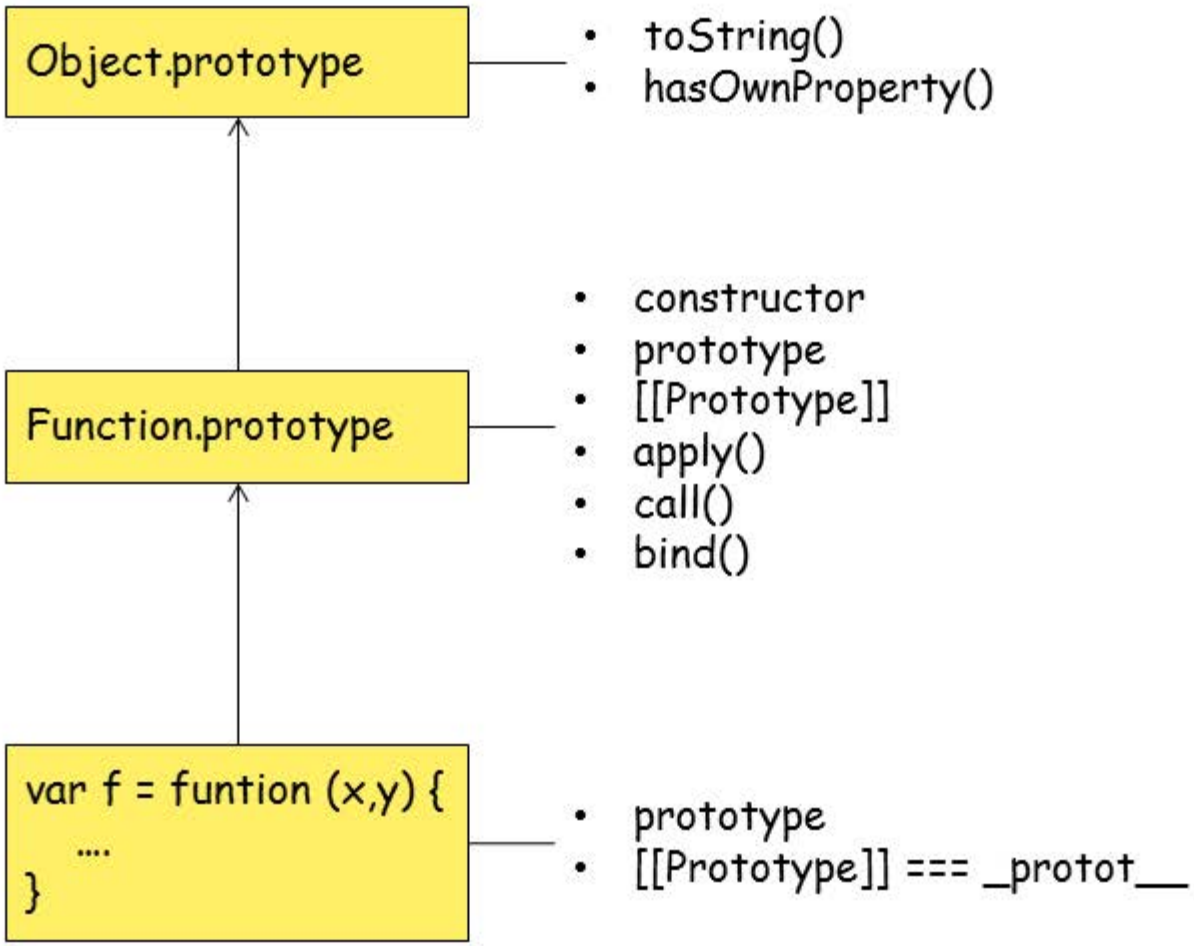
- 매개변수로 함수 사용 가능
- 리턴값으로 함수 사용 가능(클로저)

```
<script>
    // 변수에 함수 대입
    var plus = function (a, b) {
        return a + b;
    };
    console.log(plus(1,2) ) ;

    //프로퍼티에 함수 대입
    var obj = {
        add : function (a, b) {
            return a + b;
        }
    };
    console.log( obj.add(2, 3) ) ;

    //배열 요소로 함수 대입
    var arr = [ 273, plus ];
    console.log( arr[1](3, 4) );
    // .. 뒷장에
```

```
    // 앞장에 이어서..
    // 매개변수로 함수 사용
    var alarm = function ( func ) {
        console.log('alarm start');
        func();
        console.log('alarm end');
    };
    alarm(
            function () {
                console.log('매개변수로 함수 사용');
            }
    );

    // 리턴값으로 함수 사용 : 클로저
    var doWork = function (x) {
        return function calculate(y) { return x + y; };
    }
    var func = doWork(5);
    console.log( func(6) );
</script>
```

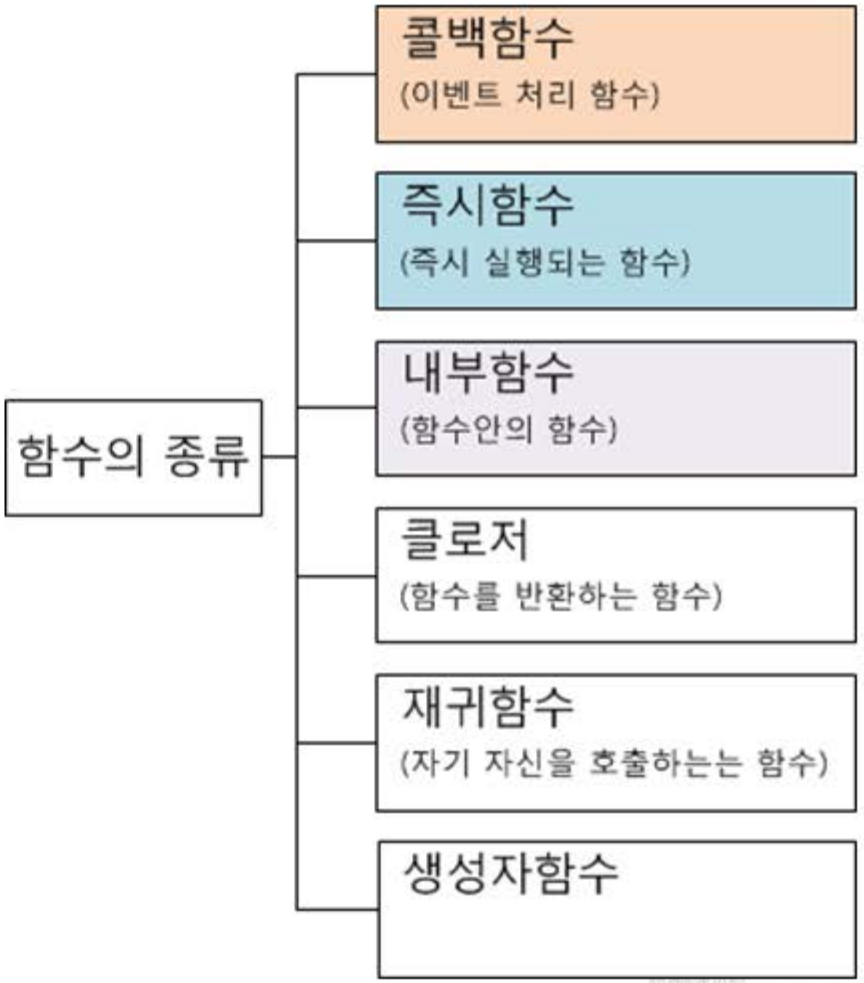
```
<script>
    // 매개변수로 함수 사용
    var alarm = function ( func ) {
        console.log('alarm start');
        func();
        console.log('alarm end');
    };

    var f = function ( ) {
        console.log('매개변수로 함수 사용');
    };

    alarm(  ?  );  // f  or  f()
</script>
```

함수 계층도 - prototype 체인



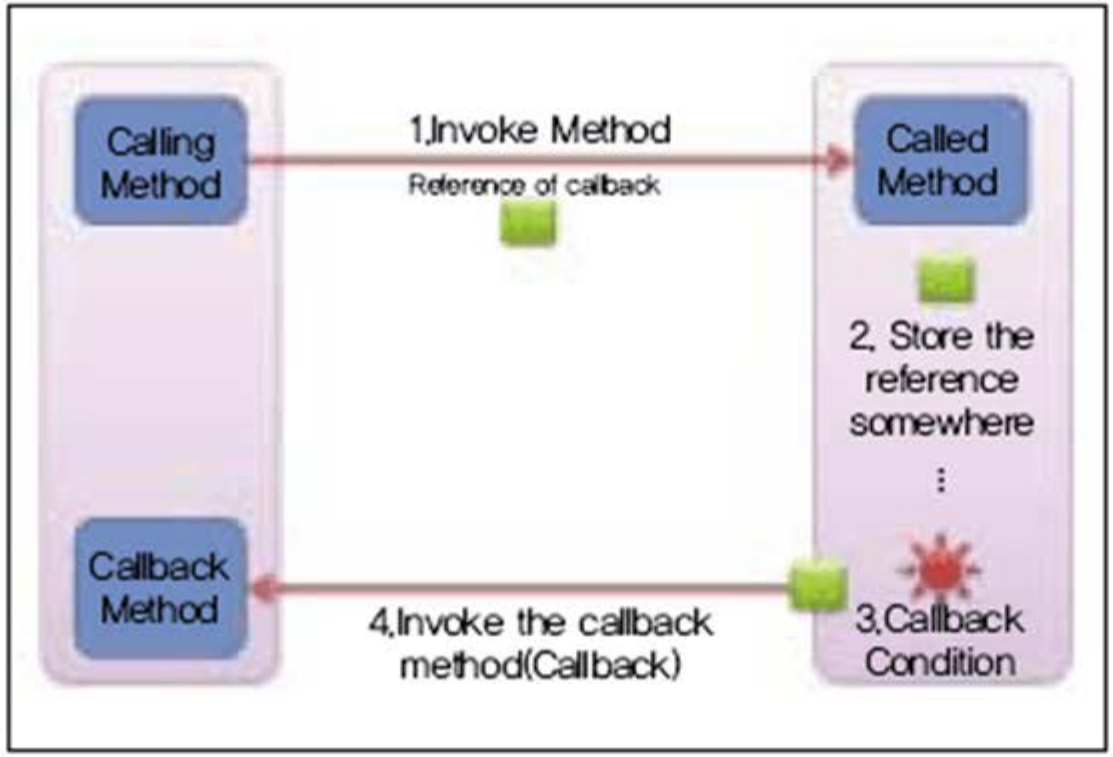
함수 종류

- 콜백 함수 - 이벤트 처리
- 즉시 함수
- 내부 함수
- 클로저 함수
- 재귀 함수(하노이타워)
- 생성자 함수



함수 종류 - 콜백 함수

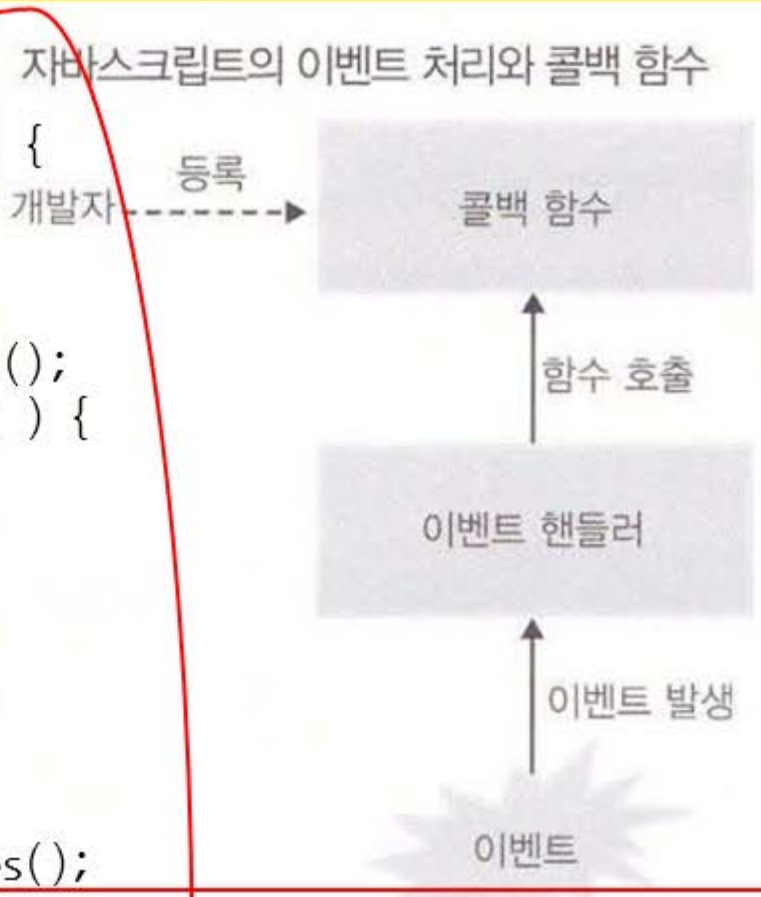
- 콜백 함수란
  - 이벤트가 발생 하였을 때 동작하는 함수
  - ex) 도난 경보 장치, 화재 경보기
- 이벤트 처리, Ajax



```
<head>
    <script>
        // 페이지 로딩되고 호출되는 콜백 함수
        window.onload = function () {
            var mesg = 'welcome!!!';
            console.log( mesg );
            alert( mesg );
        }
    </script>
</head>
<body>
    <script>
        console.log( 'before </body>' );
    </script>
</body>
```

매시간 1분마다 알림을 발생하는 콜백 함수

```
<script>
    var alarm = function (min, callback ) {
        var isCall = false;
        setInterval( function() {
            var date = new Date();
            var nowMin = date.getMinutes();
            if( nowMin == min && !isCall ) {
                //isCall = true;
                callback(nowMin);
            }
        }, 1000);
    }

    window.onload = function() {
        var curMin = new Date().getMinutes();
        alarm( curMin+1, function(min) {
                var msg = "현재는'" + min + "' 분입니다.";
                window.alert(msg);
            }
        );
    };
</script>
```



함수 종류 - 즉시 함수

- 함수를 정의함과 동시에 바로 실행되는 함수
- 초기화 작업 시 주로 사용된다
- ex) jQuery

```
var foo = function (name) {
    var mesg = 'function  --> ' + name;
    console.log( mesg );
    alert( mesg);
};
foo('foo');
```

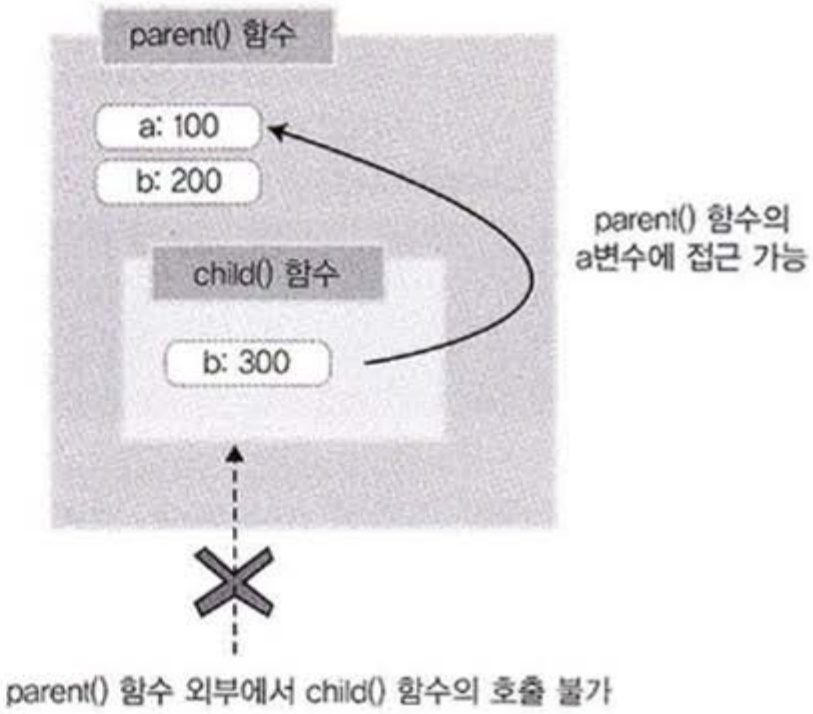
```
// 즉시 실행되는 함수
( function (name) {
    var mesg = 'immediate function  --> ' + name;
    console.log( mesg );
    alert( mesg);
})('foo');
```

```
<script>
    var foo = function (name) {

        var mesg = 'This is the immediate function  --> ' + name;

        console.log( mesg );
        alert( mesg);
    };
    foo('bar');

    // 즉시 실행되는 함수
    ( function (name) {

        var mesg = 'This is the immediate function  --> ' + name;

        console.log( mesg );
        alert( mesg);
    })('foo');
</script>
```

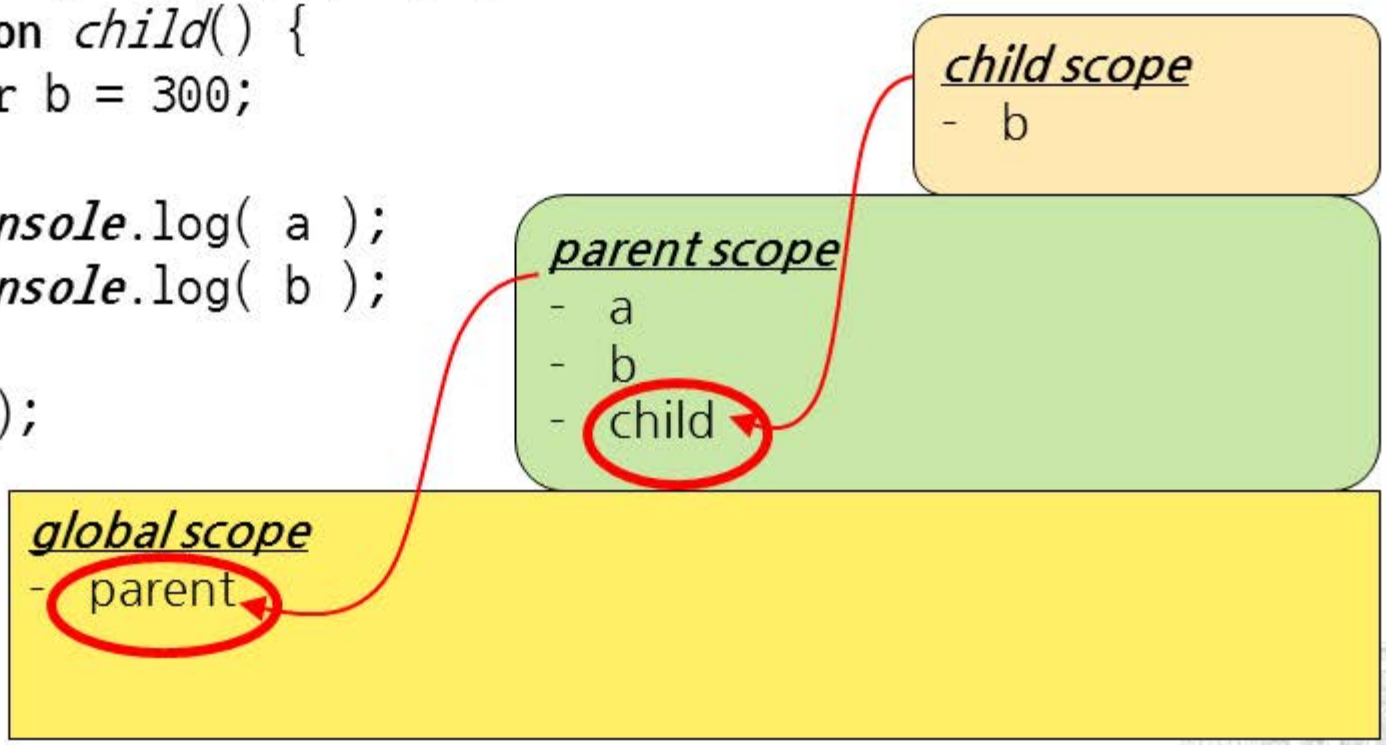
함수 종류 - 내부 함수

- 함수 안의 함수
- 내부 함수는 외부 함수의 변수에 접근 가능하다 - 스코프 체이닝
- 내부 함수는 외부 함수에서만 호출이 가능하다.

```
// parent() 함수 정의
function parent() {
    var a = 100;
    var b = 200;

    // child() 내부 함수 정의
    function child() {
        var b = 300;

        console.log( a );
        console.log( b );
    }
    child();
}
parent();
```



```
<script>
    // parent() 함수 정의
    function parent() {
        var a = 100;
        var b = 200;

        // child() 내부 함수 정의
        function child() {
            var b = 300;

            console.log( a );
            console.log( b );
        }
        child();
    }

    parent();
</script>
```



함수 종류 - 클로저 함수

- 함수를 리턴하는 내부 함수
- 내부 함수는 외부 함수의 지역 변수나 매개변수에 접근할 수 있다
- 내부 함수는 외부 함수의 arguments 객체는 호출할 수 없다

```
// outter() 함수 정의
function outer(x) {
    var a = 100;

    // inner() 내부 함수 정의
    return function () {
        return ++x;
    };
}

var x = -1;
var f = outer( x );
console.log( f() );
```

```
<script>
   // outter() 함수 정의
   function outer(x) {
      var a = 100;

      // inner() 내부 함수 정의
      return function () {
         return ++x;
      };
   }

   var x = 1;
   var f = outer(x);
   console.log( f() );
</script>
```

함수 종류 - 재귀 함수

- 자기 자신을 호출하는 함수
- 하노이 탑, factorial 구하기, 피보나치 수열

```
var hanoi = function(disc, src, aux, dst) {
    if( disc > 0 ) {
        hanoi(disc-1, src, dst, aux);
        console.log( 'Move disc ' + disc + ' from ' + src + ' to ' + dst );
        hanoi(disc-1, aux, src, dst);
    }
};
hanoi(3, 'src', 'aux', 'dst' );
```

```
var factorial = function factorial(i, a) {
    a = a + 1;
    if( i<2) {
        return a;
    }
    return factorial(i-1, a*i);
};
console.log( factorial(4) ); //
```

```
<script>
    var factorial = function factorial(i, a) {
        a = a || 1;
        if( i<2) {
            return a;
        }
        return factorial(i-1, a*i);
    };
    console.log( factorial(4) ); //
    console.log( factorial(4, 2) ); //
</script>
```

Tower of Hanoi