함수

- 자바스크립트에서 함수는 하나의 자료형.
- 함수() 의 형식으로 괄호를 열고 닫음으로 함수를 호출함.

함수의 선언

```
<script>
        function (){} //

        var  = function(){}; //
</script>
```

웹 브라우저는 script내부의 내용을 한 줄씩 읽기 전에 **선언적 함수부터 읽음**!

→ 선언적 함수는 함수를 호출 후 선언해도 오류 안남

→ 익명함수는 함수 호출 후 함수를 선언하면 오류 남

가변 인자 함수

- 매개변수의 갯수가 변할 수 있는 함수
- arguments : 매개변수의 배열

arguments 객체 사용 함수

```
<script>
        function sumAll(){
                var output = 0;
                for(var i=0; i<arguments.length; i++){
                        output += arguments[i];
                }
                return output;
        }

        alert(sumAll(1,2,3,4,5,6,7,8,9)); //45

</script>
```

내부 함수

- 함수 안에 함수
- 내부함수는 함수 안에서만 사용 가능

내부함수 예제 : 피타고라스

```
<script>
        function pythagoras(width, height){
                function square(x){
                        return x*x;
                }

                return Math.sqrt(square(width) + square(height));
        }
</script>
```

콜백 함수

- 자바스크립트는 함수도 하나의 자료형이므로 매개변수로 전달 할 수 있음.
- 함수를 매개변수로 전달하는 함수를 콜백 함수라고 함.

익명 함수를 매개변수로 전달

```
<script>
        function CallTenTimes(callback){
            for(var i = 0; i<10; i++){
                callback();
            }
        }
        var callback = function(){
            alert(" .");
        };

        CallTenTimes(callback); //alert 10
    </script>
```

클로저

- 지역변수를 남겨두는 현상
- 함수도 생성된 공간
- 리턴된 함수 자체
- 살아남은 지역변수 output

→ 지역변수가 함수가 실행 될 때 생성되고 종료후 사라지는 규칙은 위반함

클로저 예제

```
<script>
                function test(name){
            var output = 'Hello ' + name + '....!';
            return function (){
                alert(output);
            };
        }

        test('JavaScript')();
                //Hello JavaScript...!
</script>
```

→ 다음에 환용된 가능성이 있어서 output 변수 값은 제거하지 않고 남겨둠

다양한 함수들

- setTimeout(function, millisecond) → clearTimeout(id)
- setInterval(function, millisecond) → clearInterval(id)

setInterval()함수

```
<script>
    var intervalID = setInterval(function(){
            alert('<p>'+ new Date() + '</p>');
    }, 1000);

    //10
    setTimeout(function(){
            clearInterval(intervalID);
    },10000);
</script>
```

→ 10초동안 경고창을 계속 촌덕하는 함수

- **Nan을 확인할 때는 isNan() 함수 사용 (변수 === Nan 이렇게 사용하면 안됨)**

Number()와 parseInt() , parseFloat()의 차이

Number는 숫자만 있을 때 사용 가능, parseInt는 문자가 섞여있어도 숫자로 변환 할 수 있는 부분까지 모두 숫자로 변환

(참고로 앞부터 읽으므로 "W1000"같은 문자열은 변환하지 못함)

parseInt(), parseFloat()

```
<script>
        var won = '1000';
        var dollar = '1.4@';
        alert(Number(won)); //NaN
        alert(parseInt(won) + " : "+ parseInt(dollar)); //1000 : 1
        alert(parseFloat(won) + " : " + parseFloat(dollar)); //1000 : 1.4
                alert(parseInt("FF", 16)); //16  FF = 255
</script>
```

자바스크립트의 실행 순서

타이머 함수와 같은 웹 요청 관련 함수는 , 현재 실행중인 다른 코드의 실행이 끝나기 전에는 실행되지 않음

자바스크립트 실행순서

```
<script>
        alert('A');
        setTimeout(function(){
            alert('B');
        }, 0);
        alert('C');
 </script>
```

→ A C B 순서로 실행. 코드의 순서와 관계없이 setTimeout함수는 모든 코드가 실행 되고 나서 실행된다.

짧은 조건문을 활용한 기본 매개변수

짧은 조건문을 사용한 매개변수 설정

```
<script>
       function test(a,b,c){
            b = b || 52;
            c = c || 273;
            alert(a + " : " + b + " : " + c);
       }

           function test(a, b = 52, c = 273){ //
           alert(a + " : "+ b + " : "+ c);
       }

       test(1, 2); //1 : 2 : 273
</script>
```

→ 앞에 값이 없으면(undefined 자료형이라면) 뒤에 값으로 변수 초기화.

→ 엄청 자주 사용되는 형태!

→ function (<매개변수>, <매개변수>, <매개변수> = <값>, <매개변수> = <값>){}

화살표 함수

function(){} → ( )=>{ }

- this 키워드의 의미가 다름
    - 익명 함수 : 함수 자체에 바인딩 되어 있는 객체
    - 화살표 함수 : 전역 객체

화살표 함수

```
<script>
        const multiply = (a,b) => a * b; //  return return
        alert(multiply (1,2));
</script>
```

전개 연산자

- ...은 배열앞에 붙이는 것
- 가변 매개변수 함수를 만들 때 사용, arguments객체를 사용하는 것과 같은 기능
- But, arguments는 배열이 아닌 object(배열 유사 객체), 가변 매개변수는 완전한 배열.

전개 연산자

```
<script>
        function test(...numbers){
            alert(numbers[0]);
            alert(numbers[1]);
        }
        test(1,2,3); //1,2

        function test2(a,b, ...numbers){
            alert(numbers);
        }
        test2(1,2,3,4,5,6); //3,4,5,6

</script>
```

전개 연산자를 활용한 함수 호출

```
<script>
        function test(a,b,c,d){
            alert(`${a}:${b}:${c}:${d}`);
        }
        var array = [1,2];
        test(273, 53, ...array); //273 : 53 : 1 : 2
        test(...array, ...array); // 1 : 2 : 1 : 2

</script>
```

→ 배열은 인자로 넣을 때 사용