JavaScript

JavaScript 소개

- JavaScript는 Front-End 개발(눈에 보이는 부분을 개발)언어로 정적 웹을 동적 웹으로 구현시키는 기능을 함.
- 코드내용을 한꺼번에 번역해서 실행하는 Compiler 언어가 아닌 한 줄씩 읽어 실행하는 Interpreter 언어임 -> 속도가 느리나 개발하기 편함.
- JavaScript은 Front-End 개발언어의 대표이지만 Node.js을 사용하여 Back-End 개발도 가능함.

JavaScript 기초세팅

output.js

```
1. var mm = "javascript test"
2. document.write(mm);
```

example1.html

```
1. <html>
2. <head>
3.     <meta charset="utf-8">
4.     <title>example1</title>
5.     <script>
6.         function test() {
7.             document.write("테스트입니다.");
8.         }
9.     </script>
10.</head>
11.<body>
12.    <script src = "output.js"/></script>
13.    <script>test();</script>
14.    <p>tt값</p>
15.    <script>
16.        var tt = 34578;
17.        document.write("tt의 값은:"+tt);
18.    </script>
19.</body>
20.</html>
```

- 스크립트 코드는 html의 어느 곳이나 작성할 수 있음.
- (5) 스크립트는 순차적으로 실행됨으로 스크립트의 함수를 사용할 경우 <head>에서 선언하는 것이 일반적임.
- (12) 외부 스크립트 언어는 <script> 태크 안에 src 속성으로 외부 파일(output.js)을 연결해야 함. 이때 </script>를 꼭 닫아줘야 함.
- 줄주석은 // , 구간주석은 /* */를 사용함.

변수(Variable)

1. 변수 (variable)

- 선언

  var 변수명;

  var 변수명 = 값;

  변수명;

- var에서는 변수의 type(자료형)을 특별이 선언하지 않음. (실행중에 타입들을 자유롭게 변경이 가능하다 -> 동적타입(Dynamic Type)이라고 함)
- var을 따로 선언안해도 가능함.

```
1. <script>
2.     var string_t = "javascript 연습";   // 문자형
3.     var string_t1 = "111";              // 문자형
4.     var string_t2 = "<br>";             // 문자형 안에는 html 언어도 사용가능
5.     var number_t = 1234;                // 숫자형
6.     var Boolean_t = true;               // 논리형(true 또는 false)
7.     number_t1 = 4321;                    // var가 없어도 됨.
8.     document.write(string_t+'<br>'+string_t1+'<br>'+number_t+'<br>'+boolean_t);
9.  </script>
```

2. 상수 (constant, literal)

- 선언

  const 상수명 = 값;

- 한번 결정되면 변하지 않는 값을 상수라 함.
- 상수명은 통상적으로 대문자를 사용함 (소문자 사용해도 상관 없음).

```
1.     <script>
2.         const PI = 3.141592;
3.         radius = 3;
4.         document.write('넓이는'+(radius**2)*PI);
5.     </script>
```

3. 변수의 형 변환 (conversion)

- 암시적 형 변환(implicit conversion) : 형 변환이 매우 자유롭게 특별한 규칙이 없음.

```
1.     <script>
2.         var test = 1;
3.         aa = test;
4.         document.write(aa);
5.     </script>
```

- 명시적 형 변환(explicit conversion) : 강제적으로 형을 변환 시켜줌(캐스팅)

```
1. var aa = "42";
2. var cc = Number(aa)+4;       // Number(문자) 문자를 숫자로 변환
3. document.write(cc+'<br>');
4. var dd = String(cc)+'1';     // String(숫자) 숫자를 문자로 변환
5. document.write(dd+'<br>');
6. var ee = Boolean(0);         // Boolean(숫자 또는 기호)   Boolean 안쪽 0, "", null,
   undefined, Nan이 들어가면 false로 반환
7. var ff = Boolean("s");
8. document.write(ee + " " + ff);
```

4. 연산자

- 산술연산자 : +, -, *, /, %

```
1. document.write(4/3);                 // 실수 출력
2. document.write(4.0/3);               // 실수 출력
3. document.write(parseInt(4.0/3));     // parseInt(실수) -> 실수를 정수로 만들어 줌
4. document.write(4%3);                 // 나머지 1이 출력
```

- 증감연산자 : +=, -=, *=, /=, ++, --

```
1. a = 3, b = 5;
2. a += 1;                          // a = a + 1 과 같음.
3. b -= 1;                          // a = a - 1 과 같음.
4. document.write(a+" "+b+"<br>");
5. a *= 2;                          // a = a * 2
6. b /= 2;                          // b = b / 2
7. document.write(a+" "+b+"<br>");
8. c = a++;                             // c에 a값을 대입하고 a를 1 증가(후연산)
9. d = ++a;                             // a에 1 증가하고 d에 대입(전연산)
10.document.write(c+" "+d+" "+a);
```

- 문자결합연산자 : 문자열 + 숫자 -> 문자열숫자, 문자열1+문자열2 -> 문자열1문자열2

```
1. document.write("abc"+1+"<br>");
2. document.write("abc"+"def"+"<br>");
```

4. 연산자

- 비교연산자 : >, <, >=, <=, ==, !=, === -> true 또는 false로 답변됨.

```
1. document.write((3>4)+"<br>");
2. document.write((3==3)+"<br>");        // 3과 3이 같냐?    -> true
3. document.write((3!=3)+"<br>");        // 3과 3이 다르냐?  -> false
4. document.write('3'===3);              // '3' 타입과 3 타입이 같은가? -> false
```

- 삼항 조건 연산자 : 조건?값1:값2 -> 조건이 참이면 값1, 거짓이면 값2

```
1. document.write(3>4?"참":"거짓" + "<br>");
2. a = 6<=3?1:3;
3. document.write(a);
```

- 논리연산자 : &&(그리고), ||(또는), !(반대)

```
1. document.write((true&&true)+"<br>");      // &&는 둘다 모두 true이어야 true가 됨
2. document.write((3>5 || 10>6)+"<br>");     // ||는 둘 중에 하나만 true여도 true가 됨
3. document.write((!(3>4))+"<br>");          // !(true)는 false가 됨
```

제어문

1. 조건문(if)

- 선언 : 논리식이 true 일 경우 명령어를 실행

```
if(논리식) {
    명령어
}
```

```
1. var a = prompt('숫자를 입력하세요', 1000);
2. if(a==4) {
3.     document.write("ok"+"<br>");
4. }
5. if(true) {
6.     document.write("true");
7. }
```

2. 조건문(if - else)

- 선언 : 논리식이 true일 경우 명령어1, false일 경우 명령어2를 실행

```
if(논리식) {
    명령어1
} else {
    명령어2
}
```

```
1. num = 10;
2. if(num%2 == 0) {
3.     document.write("나머지는  0");
4. } else {
5.     document.write("나머지는 1");
6. }
```

3. 조건문(if - else if)

- 선언 : 논리식 1이 true일 경우 명령어1, 논리식 2가 true일 경우 명령어2, … 모두 아닐 경우 else 명령어 실행

```
if(논리식1) {
      명령어1
} else if(논리식2) {
      명령어2
} else if(논리식3) {
      명령어3
} else {
       명령어4
}
```

```
1. var num=10;
2. if(num == 0) {
3.     num++;
4. } else if(num == 10){
5.     num+=3;
6. } else if(num == 13) {
7.     num++;
8. } else {
9.     num = 100;
10.}
11.document.write(num);
```

4. 조건문(switch)

- 선언 : 변수 값이 case 값에 맞는것을 찾아서 실행함. 이때 case이후에는 모두 실행. default는 무조건 실행되나 생략가능. 변수의 type은 정확히 일치해야 함.

변수선언
switch(변수) {
  case 값1:
    명령1;
  case 값2:
    명령2;
  case 값3:
    명령3;
  default:
    명령4
}

```
1. var n = 3;
2. switch(n) {
3.     case 1:
4.         document.write("a");
5.     case 2:
6.         document.write("b");
7.     case 3:
8.         document.write("c");
9.     default:
10.        document.write("d");
11.}
```

4. 조건문(switch - break)

- 선언 : 변수 값이 case 값에 맞는것을 찾아서 실행하고 끝냄.

```
변수선언
switch(변수) {
   case 값1:
      명령1;
      break;
   case 값2:
      명령2;
      break;
   default:
      명령4
}
```

```
1. var inp  = "1";
2. switch(inp) {
3.     case "0":
4.         document.write("a");
5.         break;
6.     case "1":
7.         document.write("b");
8.         break;
9.     case "2":
10.        document.write("c");
11.        break;
12.    default:
13.        document.write("d");
14.}
```

5. 반복문(while)

- 선언

  변수정의

  while(논리식) {

  명령어;

  증감식;

  }

```
1. var num=0;
2. while(num<5){
3.     document.write(num+"<br>");
4.     num++;
5. }
```

6. 반복문(do - while문)

- 선언

변수정의

do {

명령;

증감식;

} while (논리식)

```
1. var num=0;
2. do {
3.     document.write("숫자는 "+num);
4.     num++;
5. } while(num<5);
```

7. 반복문(for)

- 선언

for (초기화; 논리식; 증감식) {
  명령어
}

```
1. for(i=0; i<5; i++) {
2.     document.write(i+"<br>");
3. }
4. document.write(i+"<br>");
```

```
1. for(i=0; i<5; i++) {
2.     for(j=0; j<3; j++) {
3.         document.write(i + " " + j+"<br>");
4.     }
5.     document.write("<br>")
6. }
```

```
1. for(i=5; i>0; i--) {
2.     document.write(i+"<br>");
3. }
```

```
1. out1: for(i=0; i<10; i++) {
2.     for(j=0; j<10; j++) {
3.         document.write(i+" "+j+"<br>");
4.         if(j == 3) {
5.             break out1;              // out1에
   지정되어 있는 for문까지 빠져나간다.
6.         }
7.     }
8. }
```

8. break문

- break문 : break문을 만나면 반복문을 빠져나옴.

```
1. for(i=0; i<10; i++) {
2.     if(i==2) {
3.         break;
4.     }
5.     document.write(i+"<br>");
6. }
7. for(j=0; j<5; j++) {
8.     for(k=0; k<5; k++) {
9.         if(k==2) {
10.            break;
11.        }
12.        document.write(i +" "+j+"<br>");
13.    }
14.}
```

9. continue문

- continu문 : continue문을 만나면 다음 문장이 실행안되고 다음 step으로 넘어감.

```
1. sum=0;
2. for(i=0; i<10; i++) {
3.     if(i==2) {
4.         continue;
5.     }
6.     sum++;
7. }
8. document.write(sum);
```

10. 무한루프

- 무한루프 사용은 다음과 같음.

```
for(;;) {
   명령어
}

while(true) {
 명령어
}
```

함수(Function)

1. 함수의 정의

- 함수는 자주 사용되는 코드에 이름을 붙여 정의한 객체임.
- 함수는 어느 위치에 있든 상관 없음.
- 정의

function 함수명(인수목록) {

명령어;

return 값; (return 은 옵션임)

}

```
1. function sum(a,b) {          // 인수 2개를 받는다.
2.     return a + b;            // 인수 2개 더한 것을 return 시킨다.
3. }
4. function sum1(a,b) {         // 인수 2개를 받는다.
5.     document.write(a+b);     // 인수 2개 더한 것을 출력한다.
6. }
7.
   document.write(sum(3,2)+"<br>");     // return 되어서 나온 값을 출력한다.
8. sum1(3,5);                           // sum1 함수를 실행시키고 3과 5를 보낸다.
```

2. 함수의 인수전달 방식

- call by value : 정수, 실수, 문자열 같은 type은 기본형으로 값을 전달함. 값들은 함수 내 인수들(지역변수)에 저장됨.
- call by reference : 객체는 참조로 전달함(메모리 주소를 링크시킴). 배열이 대표적이며, 함수에서 배열을 수정하면 함수 외부의 배열 내용이 바뀜.

```
1. function print(c,b) {          // c는 값이 저장되고, b에는 a리스트의
   메모리주소(참조값)가 저장됨
2.     b[c] = 3;
3. }

4. var a = [1,2,3];              // 리스트 선언
5. print(1, a);                  // 값 1과 a의 메모리 주소를 보냄
6. document.write(a[2]);         // print 함수에서 변경한 값이 출력됨.
```

3. 내부함수(Inner function)

- 함수 내에 함수를 포함하는 것도 가능함.
- 함수 내에 포함된 함수는 외부에서 사용이 불가능함.

```
1. function sum(a,b) {
2.     return a+b;
3. }
4.
   function sum1(a,b) {
5.     return sum(a,b) + b     // sum(a,b)를 사용
6. }
7.
   document.write(sum1(3,2))
```

```
1. function sum1(a,b) {
2.     function sum(a,b) {
3.         return a+b;
4.     }
5.     return sum(a,b) + b     // 내부의 sum(a,b)를 사용
6. }
7.
   document.write(sum1(3,2))
```

4. 익명함수(Anonymous function)

- 함수도 하나의 type이기 때문에 임의 변수에 저장할 수 있음.
- 임의 변수에 저장하게 되면 그 변수를 익명함수라고 함.
- 함수들은 어느 위치에 있건 상관 없지만 익명함수는 미리 선언되어야 함.

```
1. var test = function(a,b) {    // 함수의 이름이 없음(익명함수)
2.     function sum(a,b) {
3.         return a+b;
4.     }
5.     return sum(a,b) + b     // 내부의 sum(a,b)를 사용
6. }

7. document.write(test(3,9))

8. document.write(function(a,b){return a+b;}(3,4));  // 다음과 같이 함수옆에 인수들을
   삽입하여 실행도 가능
```

5. 클로져(Closure)

- 함수 안에 선언된 변수는 함수가 끝나더라도 파괴하지 않고 남겨두는 것을 의미.

```
1. function abc() {
2.     var num = 111;
3.     function def() {            // abc라는 함수 안에 def 함수가 존재함.
4.         document.write(num);
5.     }
6.     return def;                 // abc라는 함수는 def 함수를 return함
7. }
8. var ghi = abc();                // abc로부터 return된 함수(실행완료)는 ghi에 저장
9. ghi();                          // ghi를 실행하면 abc() 함수가 return 이후에  끝났음에도
   불구하고 num을 출력시킴(클로져)
```

6. 동적함수(Dynamic function)

- 미리 정해 놓은 함수가 아닌 실행 중에 생성하는 함수임.
- 정의:

var 변수 = new Function("인수", "인수", ... "함수내용")

```
1. // tmpe함수는 a, b를 받아서 d값을 return 함
2. var temp = new Function("a", "b", "c = 3; d = a+b+c; return d;");
3. // tmpe 함수를 실행
4. document.write(temp(4,2));
```

7. 재귀 함수(recursive)

- 재귀 함수는 자기 자신을 실행하는 함수형태를 갖음.
- 재귀 함수는 스택(가장 끝의 기억장소, 지역변수와 매개변수가 저장되는 공간) 기억장소에 저장됨.

```
1. function factorial(n) {
2.     if(n==1) {
3.         return 1;
4.     } else {
5.         return n * factorial(n-1);
6.     }
7. }

8. document.write(factorial(4));
```

객체

1. 개요

- C++, JAVA와 같이 완벽한 객체 지향 언어는 아니지만 비슷한 형태를 갖고 있음.
- C 언어의 구조체와 유사함.
- 다음 4가지 객체로 분류가 됨
  - 원시객체 : javascript 언어가 제공하는 객체
  - BOM 객체 : 브라우저가 제공하는 객체
  - DOM 객체 : 문서의 구조를 읽기 위한 객체
  - 사용자 정의 객체 : 사용자가 필요로 해서 정의한 객체
- 정의:

var 이름 = { 맴버:값, 맴버:값 .... }; // 여기서 맴버는 속성과 메소드를 포함.

2. 객체의 속성 참조, 삽입, 삭제

- 속성의 참조

객체.속성;

객체["속성"];

- 속성의 삽입

객체.새로운속성 = 값;

객체["새로운속성 "] = 값;

- 속성의 삭제

delete(객체.속성);

```
1. var object_test = {
2.     name : 'abc',       // name이라는 key에 'abc'라는 value을 갖음.
3.     number : 123,       // number라는 key에 123이라는 value을 갖음.
4.
       number1 : 145,
5.     number2 : 257,
6.     number3 : 325
7. };
8.
   document.write(object_test.name + ' '
   +object_test.number+'<br>');          // 객체이름.속성
9. document.write(object_test['name'] + ' '+ object_test["number"]
   + '<br>');  // 객체이름['속성']  도 가능
10.
   // 삽입
11.object_test.number4 = 421;      // number4 key에 대한 421 value
12.object_test['number5'] = 789;  //

13.// 삭제
14.delete(object_test.number4)          // number 속성을 삭제
15.

   for(i=1; i<6; i++) {
16.    document.write(object_test['number'+i]+'<br>');
                // ['속성'+i]를 통하여 key 값들을 지정할 수 있음.
17.}
```

3. 객체의 객체, 메소드

- 객체 안에 또다른 객체를 생성할 수 있음.

정의 :

var 객체 = {

    객체 : {

    }

};

- 객체 안에 있는 함수를 메소드라 함.

var 객체 = {

    메소드이름 : function(매개변수..) {

    }

};

```
1. var test = {
2.     name : 'abc',
3.     number : 123,
4. // 객체 안에 객체 생성 가능
5.     test1 : {
6.         name : 'abc-1',
7.         number : 345
8.     },
9. // 객체 안에 함수(메소드)를 추가
10.    test3 : function(a,b) {
11.        c = a+b;
12.        return c;
13.    }
14.};
15.// 객체안에 객체를 추가
16.test.test2 = {name:'142'};
17.// test객체 안의 test1 객체 안의 name 속성값
18.document.write(test.test1.name+'<br>');
19.document.write(test.test2.name+'<br>');
20.// test3 메소드를 실행
21.document.write(test.test3(3,4));
```

4. 객체의 with

- with 를 사용하면 맴버 앞 객체이름을 붙일 필요가 없음.
- 정의:

with(객체) {

맴버1 ;

맴버2 ;

}

```
1. var test = {
2.     name : 'def',
3.     number : 123,
4.     test1 : {
5.         name : 'abc-1',
6.         number : 345
7.     }
8. };

9. with(test) {                                  // test 객체 안의 맴버(속성, 메소드)를 with
10.    document.write(name+'<br>');               // name앞에 test. 를 붙일 필요가 없음.
11.    document.write(test1.name+'<br');
12.}
```

5. 객체의 참조

- 객체를 다른 변수에 저장할 수 있으며, 이때는 값이 저장되는 것이 아닌 참조를 함.

```
1. var test = {
2.     name : 'def',
3.     number : 123,
4.     test1 : {
5.         name : 'abc-1',
6.         number : 345
7.     }
8. };

9. var test2 = test;       // test의 값 아닌 참조 형태로 test2에 대입

10.document.write(test.name+'<br>');        // test안의 name 출력
11.test2.name = 'abc';
12.document.write(test2.name+'<br>');       // test안의 name 출력
```

6. 클래스(생성자)

- 생성자(constructor)는 객체를 만드는 역할을 하는 함수임.
- new 키워드를 사용하여 함수를 객체화 시키고, 여러개 객체들은 독립적으로 사용할 수 있음(java 클래스의 instance 의미)
- 함수 안의 모든 변수, 함수(내부함수)들 이름 앞에는 this 를 붙여야 함.
- 정의:

```
function 함수이름(매개변수) {
  this.함수이름 = function(매개변수) {
      ....
  }
  this.함수이름 = function(매개변수) {
      ....
  }
}
```

객체화 → var 변수 = new 함수이름(값);

6. 클래스(생성자)

```
1. function test(c) {                // test라는 함수는 c를 받음
2.     this.num = c;                 // this를 붙임.
3.     this.test1 = function() {     // 내부함수 정의와 다르게 변수 = function() { }을 사용함.
4.         a = 3;
5.         b = 4;
6.         return a+b+this.num;
7.     };
8. }

9. function test3() {                // 또 다른 함수 test3
10.    this.aa = 4;
11.}

12.var tt = new test(10);            // test라는 함수를 new를 통하여 tt 객체로 저장
13.document.write(tt.test1()+'<br>');  // test라는 함수에 test1을 실행시킴.
14.document.write(tt.num+'<br>');      // 함수가 객체화 되었을 경우 함수 내부 변수들도 참조가 가능.
15.var ttt = new test(7);           // test라는 함수를 new를 통하여 ttt 객체로 저장 tt와는 독립적임.
16.document.write(ttt.test1()+'<br>');  // test라는 함수에 test1을 실행시킴.

17.var tt1 = new test3();            // test3라는 함수를 tt1으로 객체화 시킴.
18.document.write(tt1.aa);
```

7. 함수의 프로토타입(prototype)

- new를 통하여 함수를 여러개 객체화 시킬 때 용량이 큰 내부 함수가 있다면 각 new 객체마다 메모리 증가가 비효율적으로 증가함.
- 이에 공유할 수 있는 내부 함수를 prototype으로 지정함.

```
1. function test(c) {                  // test라는 함수는 c를 받음
2.     this.num = c;                   // this를 붙임.
3. }
4. test.prototype.test1 = function() {   // test라는 함수의 prototype을 붙여서 test1 함수를 정의(공유함수)
5.     a = 3;
6.     b = 4;
7.     return a+b+this.num;
8. };

9. var tt = new test(10);              // test라는 함수를 new를 통하여 tt 객체로 저장
10.document.write(tt.test1()+'<br>');  // test라는 함수에 test1을 실행시킴.
11.var ttt = new test(7);             // test라는 함수를 new를 통하여 ttt 객체로 저장 tt와는 독립적임.
12.document.write(ttt.test1()+'<br>');  // test라는 함수에 test1을 실행시킴.
```

원시객체

1. 원시객체(Object)

- 언어가 자체적으로 제공하는 객체로 가장 기본이 됨.
- new 연산자를 사용하는 것이 원칙이지만 Object는 생략해도 됨.
- 대표메소드

  - toString() : 객체를 문자열로 표현

  - valueOf() : 원시값을 추출

```
1. var a = new Object();           // object 객체를 생성
2. var b = new Object(3213);       // object 객체 생성과 동시에 값 지정
3. var c = new Object('sfwe');     // object 객체 생성과 동시에 값 지정
4. var a1;                         // object 생략가능
5. var b1 = 3213;                  //
6. var c1 = 'sfwe';
document.write(typeof(a)+' '+a+'<br>');               // a의 타입은 object
document.write(typeof(b)+' '+b.toString()+'<br>');    // b가 갖는 숫자를 string으로 변환
document.write(typeof(c)+' '+c+'<br>');               // c의 타입은 object
document.write(typeof(a1)+' '+a1+'<br>');             // a1은 타입이 없음
document.write(typeof(b1)+' '+b1+'<br>');             // b1은 number 객체
document.write(typeof(c1)+' '+c1+'<br>');             // c1은 string 객체
```

2. 원시객체(Number)

- 숫자형을 표현하는 객체, Number는 생략이 가능
- 대표메소드:

- toExponential(x) : 부동소수점 형식 표현, x는 10진수로 소수점 이하 몇자리까지?
- toFixed(x) : 고정소수점 형식 표현
- toPrecision(x) : 지정한 유효숫자까지 변환

```
1. var a = 1234;                    // 숫자를 저장
2. var b = new Number(1234);        // Number 객체를 사용하여 숫자 1234 저장
3. var c = new Number("1234");      // 문자열을 숫자 1234저장
4. var d = Number("1234");          // 문자열을 숫자 1234저장

5. a += 1;      // 1235
6. b += 1;      // 1235
7. c += 1;      // 1235
8. d += 1;      // 1235

9. document.write(a+ ' '+b+' '+c+' '+d+'<br>');
10.document.write(a.toExponential(3)+'<br>');      // 1.235e+3
11.document.write(a.toFixed(3)+'<br>');            // 1235.000
12.document.write(a.toPrecision(3)+'<br>');        // 1.24e+3
```

3. 원시객체(Boolean)

- Boolean은 별도의 속성이나 메소드가 없음.

```
1. var a = new Boolean(true);
2. var b = new Boolean(1);
3. var c = new Boolean(3);
4. var d = new Boolean(0);      // false
5. var e = Boolean(null);       // new 생략가능 -> false

6. document.write(a+"<br>");
7. document.write(b+"<br>");
8. document.write(c+"<br>");
9. document.write(d+"<br>");
10.document.write(e+"<br>");
```

4. 원시객체(String)

- String은 문자열을 표현하는 객체

  var a = new String("abc"); 또는 var a = "abc"; 로 표현함.
- String 객체의 속성은 문자열 길이를 나타내는 length 밖에 없음. 메소드는 많음.
- 다음과 같은 대표적인 메소드가 있음.

메소드	설명
charAt(index)	index 위치의 문자를 반환
indexOf(item, start)	start 이후에 item이 위치해 있는 index를 반환
concat(문자열,문자열…)	문자열들을 서로 합침.
toLowerCase()	소문자로 변환
toUpperCase()	대문자로 변환
replace(old, new)	old 문자열을 new 문자열로 고치고 반환
substring(start, end)	start부터 end까지의 문자열을 추출
substr(start, length)	start에서 시작하여 length 길이 만큼 문자열을 추출
split('구분자')	문자열을 구분자에 따라 분리하여 배열로 다시 반환

배열(array)

1. 배열선언

- 배열은 여러 자료들을 하나의 이름으로 모아 놓은 객체임.
- 정의: 배열명 = [값, 값 , ...] ex) var a = [123, '123', [1,2,3]]
- 특징:
  - 다른 타입의 값들이 들어갈 수 있음. (또 다른 array 도 가능)
  - 배열의 크기가 가변적임. a=[]; 이후에 a[3] = 4; 라고 하면 0,1,2번째는 undefined가 들어가고 3번째는 4가 들어감.
  - 중간에 공백을 넣는 형태도 가능. a=[1,2,3,,,10]에서 a[3], a[4]는 undefined가 됨.
  - 배열 요소 값 삭제 가능. delete(a[2]) 라고하면 index 2번째가 undefined가 됨. 다른 요소의 index는 영향을 안 받음(index가 shift 안됨.).
  - key-value 를 통하여 값을 저장할 수 있음. a['abc'] = 'def'에서 key는 'abc', value는 'def'. (배열도 일종의 객체이며 a[첨자] 에서 첨자는 객체의 속성임(객체 속성 페이지 볼 것).
  - length : 기존 10개의 요소를 갖는 배열이라고 할 때, 배열명.length = 3; 과 같이 지정하면 나머지 7개가 삭제되고, 배열명.length=20;과 같이 지정하면 기존보다 10개가 증가됨.

2. for in 문

- 배열의 첨자나 객체의 속성을 순회하며 루프를 제어.
- 정의:

  for (변수 in 배열){

  배열[변수];

  }
- 배열의 첨자를 변수에 저장하고 명령어를 실행.

```
1. var a = [];                  // 빈 배열 a 정의
2. b = [1,2,3];                 // 배열 b 정의
3. a[0] = b;                    // 0번째 index에 b 배열을 삽입
4. a[2] = 'abc';                // 1번째 index에 'abc' 문자열을 삽입  a = [[1,2,3], 'abc']
5. a['문자도됨'] = '정말?';      // '문자도됨' 키의 값은 '정말?'
6. delete(a[0][1]);                // 2가 undefined 됨.
   a.length = 10;               // a 요소 총 갯수를 10으로 정함.
   for(i=0; i<a.length; i++){           // 10개까지 출력함.
7.     document.write(a[i]+"<br>");
8. }
   document.write("<br>");
   for(i in a) {                        // a의 첨자만 i에 저장해서 출력함.
9.     document.write(a[i]+"<br>");
10.}
```

3. 메소드

- indexOf(item, start) : 배열에서 item이 어느 index에 있는지 반환. start는 어느 위치부터 검색할 것인지 정함.

```
1. a = [4,2,1,5,1,4];
2. document.write(a.indexOf(4));       // 0
3. document.write(a.indexOf(4,2));     // 5
```

- push(item, item, item ..) : 배열의 끝에 요소를 추가. 복수가 가능함.

```
1. a = [4,2,1,5,1,4];
2. a.push(100,20,40);
3. document.write(a);       // [4,2,1,5,1,4,100,20,40]
```

- pop() : 배열의 끝에 요소를 제거함.

```
1. a = [4,2,1,5,1,4];
2. a.pop();
3. document.write(a);       // [4,2,1,5,1]
```

3. 메소드

- shift() : 배열 처음의 요소를 제거함.

```
1. a = [4,2,1,5,1,4];
2. a.shift();
3. document.write(a);       // [2,1,5,1,4]
```

- unshift(item, item, ...) : 배열 처음의 요소에 추가함.

```
1. a = [4,2,1,5,1,4];
2. a.unshift(5,9,0);
3. document.write(a);       // [5,9,0,2,1,5,1,4]
```

- reverse() : 배열을 거꾸로 정렬

```
1. a = [4,2,1,5,1,4];
2. a.reverse();
3. document.write(a);       // [4,1,5,1,2,4]
```

- sort() : 배열을 오름차순으로 정렬

```
1. a = [4,2,1,5,1,4];
2. a.sort();
3. document.write(a);       // [1,1,2,4,4,5]
```

3. 메소드

- slice(start, end) : start부터 end까지(end는 포함 안됨) item을 떼어내어 새로운 array 생성.

```
1. a = [4,2,1,5,1,4];
2. b = a.slice(1,3);
3. document.write(b);        // [2,1]
```

- concat(배열1, 배열2 ...) : 여러개의 배열을 합침.

```
1. a = [4,2,1];
2. b = [3,2,1];
3. c = [1,2,3];
4. d = a.concat(b,c);
5. document.write(d);        // [4,2,1,3,2,1,1,2,3]
```

- join(형태) : 배열의 item들을 형태와 합쳐서 문자열로 생성.

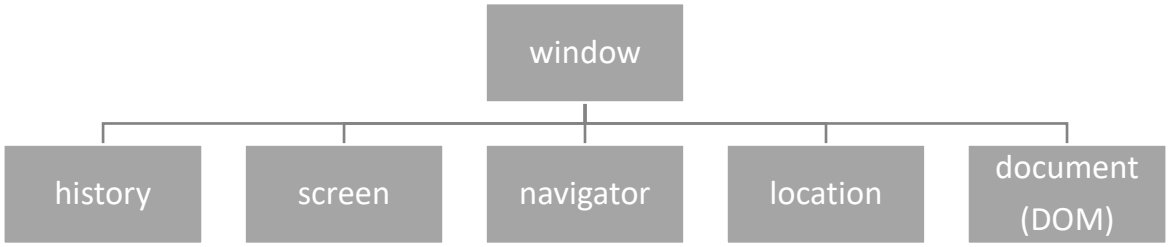
```
1. a = [4,2,1];
2. b = a.join('$');
3. c = a.join('');
4. document.write(b+"<br>");        // 4$2$1
5. document.write(c+"<br>");        // 421
```

- 문자열도 1차 배열이라고 보기 때문에 지금까지 설명한 메소드가 대부분 실행이 됨. 그러나 문자열을 수정하는 것은 안됨.

BOM
(Browser Object Model)

1. BOM 설명

- 브라우저의 정보를 읽거나 조작하는 기능을 제공.
- 브라우저마다 차이가 있음(표준화가 안되어 있음).



- window: 최상위 계층의 root 객체임. 소속되어 있는 속성이나 메소드는 window. 생략가능
- history: 사용자가 방문했던 웹 사이트 목록을 저장하고 관리하는 객체
- screen: 컴퓨터 화면에 대한 정보를 제공하는 객체
- navigator: 브라우저와 운영체제에 대한 정보를 제공하는 객체
- location: URL과 같은 네트워크상의 정보들을 제공하는 객체
- document: HTML 문서를 읽고 조작하는 객체 (DOM이라고도 함)

2. BOM(window)

- 최상위 root 객체로 다음과 같은 속성과 메소드를 제공함.
- window. 은 생략해도 됨.

속성	설명
name	윈도우의 이름
closed	윈도우가 닫혔으면 true
innerWidth, innerHeight	내용 영역의 폭과 높이
outerWidth, outerHeight	툴바와 스크롤바를 포함한 폭과 높이
screenX, screenY	화면에 대한 윈도우 좌표
screenLeft, screenRight	화면에 대한 윈도우 좌표

```
1. // window. 속성
2. document.write(name);
3. document.write(innerWidth+" "+innerHeight+"<br>");
4. document.write(outerWidth+" "+outerHeight+"<br>");
5. document.write(screenX+" "+screenY+"<br>");
6. document.write(screenLeft+" "+screenTop+"<br>");
```

2. BOM(window)

메소드	설명
open("URL", "새창이름", "새창 옵션")	URL 주소 페이지를 새창을 통해 나타냄
close()	현재 윈도우를 닫음
alert(data)	경고창을 나타내고 data를 보여줌. 이후 다음 코드를 수행
prompt("질문", "답변")	질문과 답변에 대한 창을 나타냄 ("답변"이 return 됨)
confirm("내용")	내용에 대한 확인을 누르면 true가 return
moveTo(x,y)	윈도우를 x,y 위치로 이동
moveBy(x,y)	현재위치를 기준으로 상대적 x,y 이동
resizeTo(width, height)	지정한 새 창의 크기를 변경
print()	프린터로 인쇄함.
setInterval(function() { 코드 }, 시간간격)	무한번 시간간격으로 함수를 호출하여 실행
setTimeout(function(){코드}, 시간간격)	단한번 시간간격으로 함수를 호출하여 실행
clearInterval(참조변수)	setInterval을 취소함.

2. BOM(window-open)

- Open 메소드는 지정한 URL 페이지를 새 브라우저 창에 나타낼 수 있음.

정의 : window.open(“URL”, “창이름”, “창옵션”)

- URL: 웹페이지 주소.
- 창이름: 윈도우의 고유한 이름으로 부모나 다른 윈도우와 통신할 때 쓰임. 참조할 일이 없으면 굳이 안써줘도 상관없음.
- 창옵션: 다음과 같은 종류가 있음.

속성	설명
width	폭 설정
height	높이 설정
left	X축 위치
top	Y축 위치
scrollbars	스크롤바 숨김 = no, 노출 = yes
location	url 주소 입력 숨김 = no, 노출 = yes
status	상태표시줄 숨김 = no, 노출 = yes
toolbars	도구 상자 숨김 = no, 노출 = yes
resizable	크기 조정 불가능=no, 가능=yes
close	닫기 버튼 표시 불가능 = no, 가능=yes

2. BOM(window 예제)

```
1. <script>
2. var global_var;        // 추가 창에 대한 객체 저장 변수
3. function openwindow() {     // 새로운 창 생성
4.     global_var = window.open("", "", "width=400, height=300,
   resizeable=yes");  // URL 주소를 넣으면 resize, move가 안됨.
5. }
6. function closewindow() {    // 생성된 창 닫음
7.     global_var.close();
8. }
9. function alertwindow() {
10.    window.alert("경고입니다.");
11.}
12.function promptwindow(){
13.    temp = window.prompt("질문입니다.","답변입니다.");
14.    document.write(temp);
15.}
16.function confirmwindow() {
17.    document.write(confirm("확인?"));
18.}
19.function resizeTowindow() {
20.    global_var.resizeTo(500,500);
21.}
22.function moveTowindow() {
23.    global_var.moveTo(100,100);
24.}
25.function printwindow() {
26.    window.print();
27.}
28.</script>
29.<input type="button" value="open" onclick="openwindow()"/>
30.<input type="button" value="close" onclick="closewindow()"/>
31.<input type="button" value="alert" onclick="alertwindow()"/>
32.<input type="button" value="prompt" onclick="promptwindow()"/>
33.<input type="button" value="confirm" onclick="confirmwindow()"/>
34.<input type="button" value="resize" onclick="resizeTowindow()"/>
35.<input type="button" value="moveto" onclick="moveTowindow()"/>
36.<input type="button" value="print" onclick="printwindow()"/>
```

2. BOM(window - setInterval, clearInterval)

```
1. <script>
2. var windowopen;
3. function openwindow() {          // 새창을 생성함.
4.     windowopen = window.open("","","width=200, height=200");
5. }

6. var windowclose;
7. function closewindow() {
8.     windowopen.close();          // 창을 닫음.
9. }

10.var a = 0;
11.function abc() {
12.    windowopen.document.write(a+"<br>");    // 1씩 증가된 a를 출력
13.    a++;
14.}

15.var setint;
16.function start() {
17.    setint = setInterval("abc()", 100);    // abc 함수를 100/1000초 간격으로 실행하였을 경우
18.}

19.function cancel() {
20.    clearInterval(setint);                  // 취소
21.}
22.</script>

23.<input type="button" value="창열기", onclick="openwindow()"/>
24.<input type="button" value="창닫기", onclick="closewindow()"/>
25.<input type="button" value="시작", onclick="start()"/>
26.<input type="button" value="취소", onclick="cancel()"/>
```

2. BOM(window - setInterval, clearInterval)

```
1. <script>
2. var windowopen;
3. function openwindow() {          // 새창을 생성함.
4.     windowopen = window.open("","","width=200, height=200");
5. }
6. var windowclose;
7. function closewindow() {
8.     windowopen.close();          // 창을 닫음.
9. }

10.var a = 0;
11.var setint;
12.function start() {
13.     setint = setInterval(function () {      // 다음과 같이 function을 setInterval에 직접 삽입가능
14.     windowopen.document.write(a+"<br>");    // 1씩 증가된 a를 출력
15.     a++;
16.     }, 100);    // abc 함수를 100/1000초 간격으로 실행하였을 경우
17.}

18.function cancel() {
19.     clearInterval(setint);                  // 취소
20.}
21.</script>

22.<input type="button" value="창열기", onclick="openwindow()"/>
23.<input type="button" value="창닫기", onclick="closewindow()"/>
24.<input type="button" value="시작", onclick="start()"/>
25.<input type="button" value="취소", onclick="cancel()"/>
```

2. BOM(window - setTimeout, clearTimeout)

```
1. <script>
2.     var windowopen;
3.     function openwindow() {          // 새창을 생성함.
4.         windowopen = window.open("","","width=200, height=200");
5.     }
6.
7.     var windowclose;
8.     function closewindow() {
9.         windowopen.close();          // 창을 닫음.
10.    }
11.
12.    var a = 0;
13.    function abc() {
14.        windowopen.document.write(a+"<br>");    // 1씩 증가된 a를 출력
15.        a++;
16.    }
17.
18.    var settime;
19.    function start() {
20.        settime = setTimeout("abc()", 5000);    // 5초 후에 abc 함수를 실행
21.    }
22.    function cancel() {
23.        clearTimeout(settime);                  // 취소
24.    }
25.</script>
26.
27.<input type="button" value="창열기", onclick="openwindow()"/>
28.<input type="button" value="창닫기", onclick="closewindow()"/>
29.<input type="button" value="시작", onclick="start()"/>
30.<input type="button" value="취소", onclick="cancel()"/>
```

2. BOM(history)

- history 객체는 사용자가 방문했던 웹 사이트 목록을 저장하고 관리함.
- 보안 때문에 사용자가 방문한 웹 사이트 목록을 제공하는 기능은 없음.
- 다음과 같은 속성과 메소드가 있음.

속성	설명
length	총 history의 수

메소드	설명
back	이전 페이지로 이동
forward	다음 페이지로 이동
go(위치)	위치 -1 : 이전페이지, 1:다음페이지, 0: 현재페이지

```
1. <body>
2. <input type="button" value="전" onclick="history.back()"/>
3. <input type="button" value="후" onclick="history.forward()"/>
4. </body>
```

3. BOM(screen)

- 사용자의 컴퓨터 화면에 대한 정보를 제공

속성	설명
width, height	화면크기
availWidth, availHeight	task bar를 제외한 화면크기
availLeft, availTop	task abr를 제외한 화면 좌상단 좌표
colorDepth, pixelDepth	색상 비트수

```
1. <script>
2.     document.write( screen.width +"<br>");
3.     document.write( screen.height +"<br>" );
4.     document.write( screen.availWidth +"<br>" );
5.     document.write( screen.availHeight +"<br>" );
6.     document.write( screen.availLeft +"<br>" );
7.     document.write( screen.availTop +"<br>" );
8.     document.write( screen.colorDepth +"<br>" );
9. </script>
```

3. BOM(navigator)

- 브라우저와 운영체제에 대한 정보를 제공

속성	설명
appName	브라우저 이름
appCodeName	브라우저 계통
appVersion	브라우저 버전
userLanguage	현재 사용하고 있는 언어
platform	운영체제의 종류
cookieEnabled	쿠키를 사용할 수 있는지 여부

```
<script>
    document.write( navigator.appName +"<br>");
    document.write( navigator.appCodeName +"<br>" );
    document.write( navigator.appVersion +"<br>" );
    document.write( navigator.userLanguage +"<br>" );
    document.write( navigator.platform +"<br>" );
    document.write( navigator.cookieEnabled +"<br>" );
</script>
```

3. BOM(location)

- 현재 방문중인 페이지의 네트워크 정보를 제공하는 객체

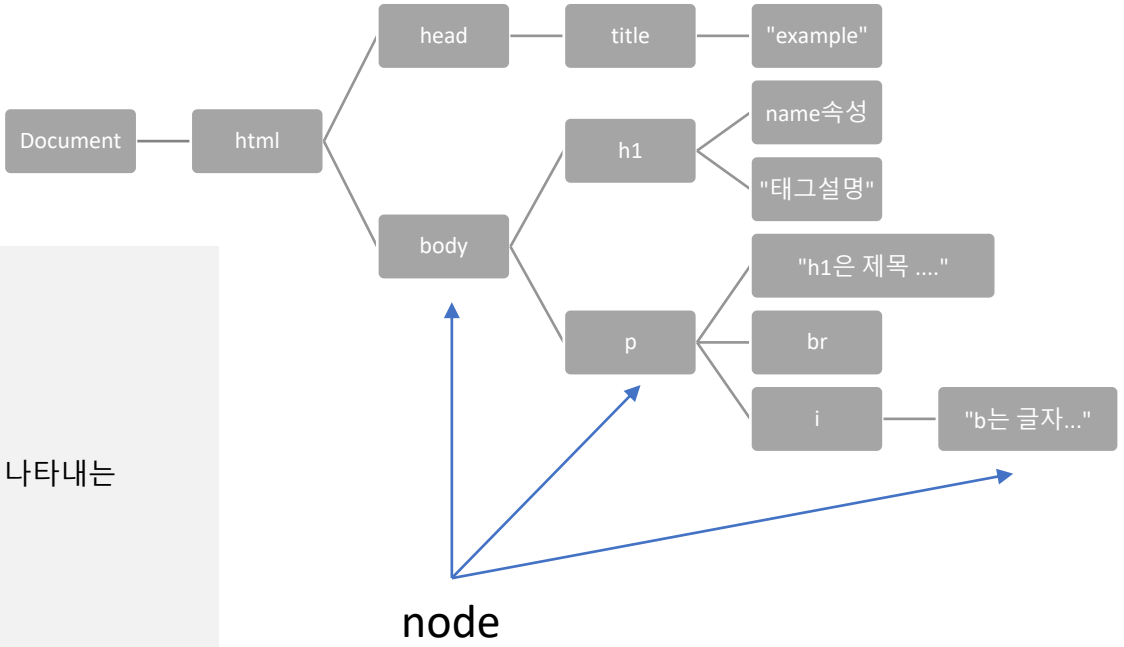
속성	설명
host	호스트 이름과 포트
hostname	호스트 이름
port	포트번호
href	전체 URL
pathname	경로
protocol	프로토콜

```
1. <script>
2.     document.write("host: "+location.host+"<br>");
3.     document.write("hostname: "+location.hostname+"<br>");
4.     document.write("port: "+location.port+"<br>");
5.     document.write("href: "+location.href+"<br>");
6.     document.write("pathname: "+location.pathname+"<br>");
7.     document.write("protocol: "+location.protocol+"<br>");
8. </script>
```

4. DOM
(Document Object Model)

1. DOM 설명

- HTML 문서를 읽고 조작하는 객체들을 모여있음 (BOM의 일부임).
- DOM은 복잡하고 거대한 문서 트리에서 원하는 정보를 쉽게 찾아 내고 조작하는 기술임.
- 최소단위 node로 계층을 이루고 있음.




```
1. <html>
2. <head>
3.     <title>example</title>
4. </head>
5. <body>
6.     <h1 name='abc'>태그설명</h1>
7.     <p>h1은 제목을 나타내는 태그이고, p는 문장을 나타내는
   태그입니다.<br>
8.         <i>b는 글자를 이텔릭체로 변경합니다.</i>
9.     </p>
10.</body>
11.</html>
```

2. Node 설명

- Document, 태그, 텍스트, 주석 모두 각각의 node임.
- Node의 속성은 DOM의 모든 객체가 공통으로 지원함.
- nodeType : 노드의 종류
  - Element : 태그를 나타냄. 1을 출력 ex) <h1> </h1>
  - Attr : 태그의 속성을 나타냄. 2를 출력 ex)<h1 id="abc"> </h1>
  - Text : 태그에 포함된 문자열 내용. 3을 출력 ex) <h1> 태그설명 </h1>
  - Comment : 주석을 나타냄. 8을 출력
  - Document : 전체 문서를 나타냄. 9를 출력
- nodeName : 노드의 이름
- nodeValue : Attr, Text, Comment 노드에서 값을 나타냄.

3. 문서에서 Node 찾기

- HTML 문서에서 Node를 찾기 위해서는 다음과 같은 방법이 있음. (찾은 후 변수에 참조됨.)
  - document.getElementById("id") : 노드에 고유한 id를 지정하였을 경우.
  - document.getElementsByName("name"): name을 통한 찾기. 이 경우 각 node마다 name들이 중복될 수 있음. 그래서 배열형식으로 리턴함.
  - document.getElementsByTagName("태그이름") : 태그 노드를 모두 찾아서 배열형식으로 리턴함. "*"라고 하면 HTML 문서 모든 태그들을 배열형식으로 리턴함.
  - document.getElementsByClassName("클래스 이름"): 태그의 class를 찾아서 배열형식으로 리턴함 (HTML5 이후부터 가능)
  - document.querySelector("CSS 선택자"): CSS 선택자를 분석하여 조건에 맞는 노드를 선택함(1개만 선택함).
  - document.querySelectorAll("CSS선택자"): CSS 선택자를 분석하여 조건에 맞는 노드를 배열형식으로 리턴함.

3. 문서에서 Node 찾기

```
1. <html>
2. <head>
3.     <title>example</title>
4. </head>
5. <body>
6.     <h1 id ="abc">태그설명</h1>
7.     <p>h1은 제목을 나타내는 태그이고, p는 문장을 나타내는 태그입니다.<br>
8.         <i>b는 글자를 이텔릭체로 변경합니다.</i>
9.     </p>
10.    <p name="pp">           <!-- name 속성은 중복이 가능함.-->
11.        또다른 문장을 쓰고 있다.
12.    </p>
13.
       <script>
14.        var find = document.getElementById("abc");
15.        document.write(find.nodeType+"<br>");   // h1에 대한 nodetype : 1(태그타입)
16.
17.        var find1 = document.getElementsByName("pp"); // pp 이름을 갖는 모든 노드를 찾음. 배열로 리턴함.
18.        document.write(find1[0].nodeType+"<br>");

19.        var find2 = document.getElementsByTagName("p"); // <p></p> 태그에 대한 노드를 모두 찾음. 배열로 리턴함.
20.        document.write(find2[0].nodeType+"<br>");
21.        document.write(find2[1].nodeType+"<br>");
22.
       </script>
23.</body>
24.</html>
```

4. 주변 Node 찾기

- 노드를 찾고 다음 속성을 사용하여 그 주변의 노드들을 찾을 수 있음.
- HTML 본문에서 줄바꿈이 있을 경우 node tree가 제대로 작동이 안될 수 있음. 모두 붙여서 써야함.

속성	설명
firstChild	첫 번째 자식 노드
lastChild	마지막 자식 노드
parentNode	부모 노드
previousSibling	이전 형제 노드
nextSibling	다음 형제 노드
ownerDocument	노드가 속한 문서의 루트 노드
childNodes	모든 차일드 노드 (배열형식으로 return됨)
attributes	속성 목록
localName	노드의 지역명
textContent	문자열로 된 내용물

5. Node 찾은 후 Element(태그) 읽기

- Element 객체는 HTML에서 태그를 표현함.
- Element 도 노드의 일종이므로 다음과 같은 속성을 갖음.
- 메소드에서는 주로 Attr(속성)을 관리하는 기능을 제공함.

속성	설명
tagName	태그의 이름
innerHTML	태그의 내용
id	id 속성
style	css 스타일

메소드	설명
getAttribute(속성이름)	이름으로부터 속성을 찾음
getAttributeNode(속성이름)	이름으로부터 속성 노드를 찾음
getElementByTagName(태그이름)	태그이름으로부터 자식 태그를 찾음
hasAttribute(속성이름)	속성이 있는지 조사함
removeAttribute(속성이름)	속성을 제거
setAttribute(속성이름, 속성값)	속성값을 변경

5. Node 찾은 후 Element(태그) 읽기

```
1. <html>
2. <head>
3.     <title>example</title>
4. </head>
5. <body>
6.     <h1 id="abc" name="ddd">테스트입니다.</h1>
7.
       <script>
8.         var test = document.getElementById("abc");
9.         document.write(test.tagName+"<br>");                // h1
10.        document.write(test.innerHTML+"<br>");              // 테스트입니다.
11.        document.write(test.id+"<br>");                     // abc
12.        document.write(test.style+"<br>");                  // CSS
13.        document.write(test.getAttribute("id")+"<br>");           //abc
14.    </script>
15.</body>
16.</html>
```

5. Node 찾은 후 Attr(속성) 읽기

- Node의 속성은 attributes 속성으로 읽으며 이는 배열로 리턴이 됨.

속성	설명
name	속성의 이름
value	속성의 값
isId	id 속성인지 조사(최신브라우져 지원안됨)
ownerElement	속성이 소속된 태그
specified	값이 지정되어 있는지 리턴

메소드	설명
getNamedItem(이름)	이름으로부터 속성을 읽음
setNamedItem(노드)	배열에 노드를 추가(기존에 있으면 덮어씀)
removeNamedItem(노드이름)	노드를 제거

```
1. <html>
2. <head>
3.     <title>example</title>
4. </head>
5. <body>
6.     <h1 id="abc" name="ddd">테스트입니다.</h1>
7.     <script>
8.         var test = document.getElementById("abc");
9.         var attr = test.attributes;     // 속성들이 배열형식으로
   return됨
10.        for(i = 0; i<attr.length; i++) {
11.            document.write(attr[i].name+" "+attr[i].value+"
   "+attr[i].isId+" "+attr[i].ownerElement+"
   "+attr[i].specified+"<br>");
12.        }
13.        document.write(attr.getNamedItem("id").value+"<br>");
     // id 속성을 찾은 다음에 값을 출력
14.        document.write(attr.getNamedItem("id").name+"<br>");
   // id 속성을 찾은 다음에 속성이름 "id"를 출력
15.    </script>
16.</body>
17.</html>
```

5. Node 찾은 후 Text 읽기

- 노드의 내용을 읽는 방법은 1. nodeValue 속성을 읽는 것임. nodeValue는 소속되어 있는 값만 읽음.
- 2. textContent 속성을 읽는 것임. textContent는 자식의 모든 노드 내용을 읽음.

```
1. <html>
2. <head>
3.     <title>example</title>
4. </head>
5. <body>
6.     <!-- 문단사이에 줄바꿈이 있으면 노드들의 tree가  -->
7.     <h1 id ="abc">태그설명</h1><p name="qq">h1은 제목을 <b>나타내는 태그이고</b>, p는 문장을 나타내는
   태그입니다.<br><i>b는 글자를 이텔릭체로 변경합니다.</i></p><p name="pp">또다른 문장을 쓰고 있다.</p>
8.     <script>
9.         var find = document.getElementById("abc");
10.        document.write(find.textContent+"<br>");            // 자식의 내용 '태그설명'
11.        document.write(find.firstChild.nodeType+"<br>");    // h1의 첫번째 자식 text의 type  -> 3
12.        document.write(find.firstChild.textContent+"<br>");   // h1의 첫번째 자식 text 내용   '태그설명'
13.        document.write(find.nextSibling.nodeType+"<br>");       // 다음 태그를 설명
14.        document.write(find.nextSibling.nodeName+"<br>");       // p
15.        document.write(find.nextSibling.textContent+"<br>");     // h1은 제목을 나타내는 태그이고, p는 문장...
16.        document.write(find.nextSibling.firstChild.nodeValue+"<br>");       // h1은 제목을
17.    </script>
18.</body>
19.</html>
```

5. Node 찾은 후 Text 변경

- 노드의 내용을 변경할 때 사용함.

```
1. <html>
2. <head>
3.     <title>example</title>
4. </head>
5. <body>
6.     <h1 id ="abc">태그설명</h1><p><b>테스트</b>입니다.</p>
7.     <script>
8.         var find = document.getElementById("abc");
9.         find.firstChild.nodeValue = "태그설명 수정";        // "태그설명" -> "태그설명 수정"으로 변경
10.        find.nextSibling.textContent = "abc";               // 테스트입니다. -> abc로 변경
11.    </script>
12.</body>
13.</html>
```

5. Node 찾은 후 CSS 변경 및 생성

- 기존 스타일 가져오기 :
  - document.querySelector("선택자") 선택된 선택자 1개만 가져옴
  - document.querySelectorAll("선택자") 모든 선택자 가져옴
- 변경 & 추가:
  - .style.속성 = "속성값" 을 통하여 변경 또는 추가 가능
- 삭제:
  - .style.removeProperty("속성") 을 통하여 기존 CSS 속성을 삭제할 수 있음.

5. Node 찾은 후 CSS 변경 및 생성

```
<html>
    <head>
        <style>
            /* 전체선택자 */
            * {background: aqua;}
            /* 태그선택자 */
            p {background: blue; color:white;}
            /* id선택자 */
            #test {background: pink; color:brown}
            /* class선택자 */
            .class_test {background:darkred; color:white}
            h3.class_1 {background: greenyellow; color:black}
            /* 그룹선택자 */
            h5, h6 {background: yellow;}
        </style>

    </head>
    <body>
        전체선택자입니다.<br>
        <p>p에 대한 text입니다.</p>
        <p id = "test">id선택자 text입니다.</p>
        <p class ="class_test">class1 선택자</p>
        <p class ="class_test">class2 선택자</p>
        <h3 class="class_1">abc</h3>
        <h5>h5테스트</h5>
        <h6>h6테스트</h6>
        <p id="insert1" style="font-size:30px;">p의 선택자</p>
        <p id="insert2">p선택자1</p>

        <script>
            function select1() {
                var aa = document.querySelector("*");
                aa.style.background="green";
            }
            // 많은 p들 중에서 1개만 가져옴
            function select2() {
                var aa = document.querySelector("p");
                aa.style.color="yellow";
            }
            function select3() {
                var aa = document.querySelector("#test");
                aa.style.color="yellow";
            }
            function select4() {
                var aa = document.querySelector(".class_test");
                aa.style.color="yellow";
            }
            // class_test를 갖고 있는 모든 클래스를 배열형식으로
            function select4_1() {
                var aa = document.querySelectorAll(".class_test");
                aa[1].style.color="yellow";
            }
            function select5() {
                var aa = document.querySelector("h3.class_1");
                aa.style.color="yellow";
            }
            // 배열 형식으로 2개 얻음.
            function select6() {
                var aa = document.querySelectorAll("h5,h6");
                aa[0].style.color="red";
                aa[1].style.color="yellow";
            }
            // 기존 폰트사이즈를 수정
            function select7() {
                var aa = document.querySelectorAll("p");
                aa[4].style.fontSize="10px";
            }
            // 없던 스타일을 생성
            function select8() {
                var aa = document.querySelectorAll("p");
                aa[5].style.background="rgb(255,255,255)";
            }
            // 위에서 생성되었는 backgound 속성을 삭제함.
            function select9() {
                var aa = document.querySelectorAll("p");
                aa[5].style.removeProperty("background");
            }

        </script>

        <input type="button" value="select1" onclick="select1()">
        <input type="button" value="select2" onclick="select2()">
        <input type="button" value="select3" onclick="select3()">
        <input type="button" value="select4" onclick="select4()">
        <input type="button" value="select4_1" onclick="select4_1()">
        <input type="button" value="select5" onclick="select5()">
        <input type="button" value="select6" onclick="select6()">
        <input type="button" value="select7" onclick="select7()">
        <input type="button" value="select8" onclick="select8()">
        <input type="button" value="select9" onclick="select9()">
    </body>
</html>
```

6. Node 추가

- 문서에 없는 태그, 속성, 텍스트, 주석 모두 추가할 수 있음. 다음과 같이 2 단계로 진행.
- 1단계: 객체를 생성
  - createElement(nodename) : 태그를 생성
  - createTextNode(text) : 텍스트 생성
    - 태그노드이름.innerHTML = "태그가 포함된 html텍스트"; innerHTML을 이용하면 태그안에 새로운 태그 삽입가능
  - createAttribute(attributename) : 속성 생성
  - createComment(text) : 주석생성
- 2단계: 생성한 노드를 트리에 삽입
  - appendChild(node) : 원하는 노드 안에서 제일 마지막에 삽입
  - insertBefore(새로운 노드, 기존노드) : 기존노드 앞에 새로운 노드가 삽입

6. Node 추가

```
1. <html>
2. <head>
3.     <title>example</title>
4. </head>
5. <body>
6.     <p>테스트입니다.</p>
7.     <script>
8.         var aa = document.createElement("p");  //1단계: 태그를 생성
9.         var bb = document.createTextNode("동적으로 생성"); // 1단계: 텍스트 생성
10.        var cc = document.createComment("주석입니다.");     // 1단계: 주석을 생성
11.        aa.appendChild(bb);                    // 2단계: 태그안에 텍스트를 삽입
12.        aa.appendChild(cc);                    // 2단계: 태그안에 주석을 삽입
13.
14.        var dd = document.createElement("p"); // 태그를 생성
15.        dd.innerHTML = "두번째 <b>동적</b>생성"; // html생성
16.
17.        document.body.appendChild(aa);         // 2단계: 새로 생성된 태그를 문서 body에 삽입
18.        document.body.insertBefore(dd, aa);    // 2번째로 생성한 태그가 첫번째 생성한 태그 앞에 위치함.
19.    </script>
20.</body>
21.</html>
```

6. Node에 속성 추가

- 태그에 속성을 추가할 수 있음.
- 추가방법은 다음과 같이 3가지가 있음.
  - 1. 속성에 직접 대입하는 방법
  - 2. 속성관련 메소드를 호출
  - 3. 속성노드를 만들어 붙이기

메소드	설명
getAttribute(속성이름)	속성을 찾아서 지정되어 있는 값을 return
setAttribute(속성이름, 값)	속성을 찾아서 값을 지정
getAttributeNode(속성이름)	속성이름을 통하여 속성노드를 찾음.
setAttributeNode(속성노드)	속성노드를 삽입함.

6. Node에 속성 추가

```
1. <html>
2. <head>
3.     <title>example</title>
4. </head>
5. <body>
6.     <p>테스트입니다.</p>
7.     <img name="newimg">
8.     <img name="newimg">
9.     <script>
10.        var oldp = document.getElementsByTagName("p");    // p는 여러개가 존재 할 수 있으니 oldp는 배열임.
11.        var createimg = document.createElement("img");
12.        var newimg = document.getElementsByName("newimg");  // name은 여러개가 존재 할 수 있으니 newimg는 배열임.
13.        // 1. 속성에 직접 대입하는 방법
14.        createimg.src = "google.png";
15.        oldp[0].appendChild(createimg);   // 기존 p에 그림을 삽입
16.
           // 2. 속성관련 메소드를 호출하여 수정(삽입)
17.        newimg[0].setAttribute("src","apple.png"); // 기존 img에 src를 변경
18.        document.write(newimg[0].getAttribute("src")+"<br>");  // apple.png가 출력
19.
20.        // 3. 속성노드를 만들어 붙이기
21.        var createattr = document.createAttribute("src"); // 속성생성
22.        createattr.nodeValue = "ms.jpg";                    // 생성된 속성에 값을 부여
23.        newimg[1].setAttributeNode(createattr);             // 기존 img 태그에 속성을 부여
24.        document.write(newimg[1].getAttributeNode("src").nodeValue+"<br>");   //ms.jpg가 출력됨.
25.    </script>
26.</body>
27.</html>
```

6. Node 삭제 및 변경

- 문서 트리에 포함된 노드를 삭제할 수 있음.
- 부모가 자식을 삭제 또는 변경 해야 함.

메소드	설명
removeChild(node)	node를 찾아서 삭제함.
replaceChile(새로운노드, 과거노드)	과거노드를 새로운 노드로 변경함

```
1. <body>
2.     <p id = "p1">테스트1 입니다.</p>
3.     <p id = "p2">테스트2 입니다.</p>
4.     <p id = "p3">테스트3 입니다.</p>
5.     <script>
6.         var test1 = document.getElementById("p1");
7.         var test2 = document.getElementById("p2");
8.         var test3 = document.getElementById("p3");

9.         // 새로운 태그 생성
10.        var createp = document.createElement("p");
11.        var createtext = document.createTextNode("교체된 text");
12.        createp.appendChild(createtext);
13.        document.body.removeChild(test2);            // body 태그 부모가 자식 p2를 삭제함.
14.        document.body.replaceChild(createp, test1);
15.    </script>
16.</body>
```

이벤트

1. 이벤트 설명

- 마우스나 키보드를 조작하여 반응을 보이는 이벤트를 만들 수 있음.
- 이벤트 발생시 실행되는 코드를 **이벤트 핸들러**라고 하고 함수형태임. 모든객체에는 이벤트가 있음.
- 아래는 대표적 이벤트로 실행시 앞에 on을 붙여서 사용해야 함. ex) onclick

이벤트	설명
load, unload	페이지 시작 및 종료
click, doubleclick	마우스 클릭, 더블클릭
mousedown, mouseup	마우스 버튼 누름, 뗌
mousemove	마우스이동
mouseover, mouseout	마우스가 경계안으로 들어 오거나 나감
contextmenu	오른쪽 마우스 클릭
submit	폼 전송
reset	폼 초기화
resize	윈도우과 프레임 크기 벼경
select	텍스트 선택

1. 이벤트 설명

```
1. <body>
2.     <h1 id="h1click">클릭1</h1>
3.     <h1 onclick="test1();">클릭2</h1>    <!-- 간단한 명령어는 이와같이 ""안에 쓸 수 있음.-->
4.     <script>
5.         var h1click = document.getElementById("h1click");
6.         h1click.onclick = function() {          // 각 객체에는 onclick 이벤트가 있고 함수로 핸들러실행
7.             alert("클릭1");
8.         }
9.
           function test1() {
10.            alert("클릭2");
11.        }
12.    </script>
13.</body>
```

2. 리스터 등록

- 핸들러를 통하여 이벤트를 실행할 수 있으나, 한번의 이벤트만 발생시킬 수 있는 단순함이 단점이 되기도 함.
- 그래서 리스너를 통하여 이벤트를 처리하는데, 리스너는 메소드를 통하여 여러핸들러를 등록하고 제거함.

메소드	설명
addEventListener(이벤트, 핸들러함수)	리스너에 등록
removeEventListener(이벤트, 핸들러함수)	리스너에서 삭제

```
1. <body>
2.     <h1 id="h1click">클릭</h1>
3.     <script>
4.         var h1click = document.getElementById("h1click");
5.         function click1() {
6.             alert("클릭1");
7.         }
8.         function click2() {
9.             alert("클릭2");
10.        }
11.        h1click.addEventListener("click", click1);
12.        h1click.addEventListener("click", click2);
13.    </script>
14.</body>
```

example

```
1. <html>
2. <head>
3.     <title>example</title>
4. </head>
5. <body>
6.
       <table id="table1" border="1">
7.         <tr>
8.             <td>abc</td>
9.             <td>def</td>
10.        </tr>
11.    </table>
12.
       <script>
13.        var table = document.getElementById("table1");
14.        function click1() {
15.            for(i=0; i<3; i++) {
16.                var row1 = document.createElement("tr");
17.                var col1 = document.createElement("td");
18.                var text1 = document.createTextNode("abc123");
19.                col1.appendChild(text1);
20.                row1.appendChild(col1);
21.                table.appendChild(row1);
22.            }
23.            // var table1 = document.getElementById("table1");
24.            // table1.setAttribute("border", "3");
25.        }
26.    </script>
27.
       <input type="button" name="click" value="click" onclick="click1()"/>
28.
   </body>
29.
30.</html>
```