16 자바스크립트와 객체

16-1. 객체 알아보기

객체란

자바스크립트에서 사용하는 객체의 종류

- **내장 객체**
  - 자바스크립트가 기본적으로 제공하는 객체로 자바스크립트 엔진에 저장되어 있음

    Date 객체, Number 객체, Array 객체, Math 객체, String 객체

- **브라우저 객체**
  - 브라우저가 제공하는 객체

    window, screen, location, history, navigator 객체 등

- **문서 객체 모델(DOM)**
  - 문서 자체는 물론 문서 내에 포함된 모든 콘텐츠를 문서 객체라고 함
  - 표준화된 방법으로 문서 객체에 접근할 수 있는 방법을 제공하기 위해 문서 객체 모델을 사용

- **자바스크립트 프로그래밍 = 기본 문법 + 내장객체 + 브라우저 객체 + DOM**

객체란

속성과 메서드의 사용 방법

- 객체와 속성, 객체와 메소드는 각각 도트문자(.)로 구분하여 표현
- 객체와 속성을 표현하는 방법

```
객체.속성 = "값"
```

[예] window.document.bgcolor="blue";

- 객체와 메서드를 표현하는 방법

```
객체.메서드( )
```

[예] window.open("test.htm", "popup", "top=50 left=50 with=300 height=100")

속성과 메소드 사용의 예

```
<form  name="login"  method="post"  action="login_proc.jsp">
     아 이 디  : <input  type=text  name="id"  size=20>
     패스워드 : <input  type=text  name ="passwd"  size=20>
     <input  type=button  value="로그인">
</form>
```

- **〈form〉 객체의 표현**
  - window.document.login
    - window : 현재 문서를 출력하고 있는 브라우저
    - document : 현재 브라우저(window)에 열려 있는 HTML 문서
    - login : 현재 HTML 문서(document)의 <form> 태그 이름

- **아이디 입력 상자**
  - 입력상자의 표현 : window.document.login.id
  - 입력 값의 표현 : window.document.login.id.value

- **아이디 입력 상자에 마우스 커서를 위치시키는 방법**
  - window.document.login.id.focus();

객체란

chapter16 > form1.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        function check() {
            if(!window.document.login.id.value) {
                alert('아이디를 입력하세요.');
                window.document.login.id.focus();
                return;
            } else {
                window.document.login.id.value="";
            }
        }

    </script>
</head>
<body>
    <form name="login" method="POST" action="#">
        아이디 <input type=text name="id" size=20>
        <input type=button value="로그인" onClick="check()">
    </form>
</body>
</html>
```

16-2. 자바스크립트의 내장 객체

내장 객체

- **자바스크립트가 기본적으로 제공하는 객체로 자바스크립트 엔진에 포함되어 있는 객체**
- **내장 객체의 생성과 사용**
  - new 연산자를 사용해 인스턴스를 생성해서 사용해야 함

```
인스턴스 =  new 내장객체
```

- **내장 객체의 종류**
  - Date 객체, Number 객체, Array 객체, Math 객체, String 객체

Array 내장 객체

배열(array)

- **프로그래밍 언어에서 사용되는 일반적인 배열의 정의**
  - 같은 자료형의 데이터가 저장되는 연속된 공간
  - 배열에 저장된 데이터들을 요소(element)라고 함
    요소에 대한 접근은 배열의 이름과 0부터 시작하는 숫자로 구성된 첨자를 사용

data[0]	data[1]	data[2]	data[3]	data[4]
10	20	30	40	50

- **자바스크립트 배열**
  - 자바스크립트에서는 서로 다른 자료형의 데이터들이 함께 하나의 배열에 저장될 수 있음

data[0]	data[1]	data[2]	data[3]	data[4]
1	홍길동	30	연구원	공학박사

Array 내장 객체

배열의 생성과 초기화

- 대괄호를 사용하는 방법

[선언]

```
배열명 = [ ];
```

```
배열명 = [ 크기 ];
```

[선언 후 초기화]

```
let arr = [ ];
arr[0] = 10;
arr[1] = 'kim;
```

```
let arr = [2];
arr[0] = 10;
arr[1] = 'kim';
```

[선언과 동시에 초기화]

```
let arr = [ 10, 'kim' ]
```

- new 연산자와 Array객체를 사용하는 방법

[선언]

```
배열명 = new Array();
```

```
배열명 = new Array(크기);
```

[선언 후 초기화]

```
let arr = new Array[ ];
arr[0] = 10;
arr[1] = 'kim';
```

```
let arr = new Array [2];
arr[0] = 10;
arr[1] = 'kim';
```

[선언과 동시에 초기화]

```
let arr = new Array( 10, 'kim' )
```

Array 내장 객체의 속성

length 속성

- 자바스크립트 Array 객체가 가지는 유일한 속성

```
배열.length
```

  - 일반적으로 배열 요소를 모두 출력할 때 순환문의 조건으로 사용

- 배열 요소의 출력
  - for 순환문이나 for – in 순환문을 사용

```
1
홍길동
30
연구원
공학박사
1
홍길동
30
연구원
공학박사
```

chapter16 > <> arrayLength.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        const user_info = [1, "홍길동", 30, "연구원", "공학박사"]

        //for 반복문 사용
        for(let i=0; i<user_info.length; i++) {
            document.writeln(user_info[i],"<br>")
        }

        //for-in 반복문 사용
        for(const i in user_info) {
            document.writeln(user_info[i],"<br>")
        }
    </script>
</head>
<body></body>
</html>
```

Array 내장 객체의 메서드

배열 요소의 결합

- concat() 메서드

```
배열3 = 배열1.concat(배열2)
```

  - 배열1의 뒷부분에 배열2를 추가하여 새로운 배열3을 생성

- join() 메서드

```
변수 = 배열.join(결합자)
```

  - 배열 요소를 인자로 [결합자]로 결합하여 새로운 문자열 생성
  - 반환 값은 배열이 아니라 문자열 임

chapter16 > concat-join.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <script>
6
7           let nums = [1, 2, 3];
8           let chars = ["a", "b", "c"];
9
10          let cnn = nums.concat(chars);
11          document.write(cnn,"<br>");
12
13          let nums_str = nums.join();
14          document.write(nums_str,"<br>");
15
16          let chars_str = chars.join('/');
17          document.write(chars_str,"<br>");
18
19      </script>
20  </head>
21  <body></body>
22  </html>
```

```
1,2,3,a,b,c
1,2,3
a/b/c
```

Array 내장 객체의 메서드

배열 요소의 삽입/삭제

- unshift() 메서드/ push() 메서드

```
배열.unshift( 배열요소 리스트 )
```

- 배열의 앞부분에 인자로 가지는 배열 요소들을 추가

```
배열.push( 배열요소 리스트 )
```

- 배열의 마지막에 인자로 가지는 배열 요소들을 추가

- shift() 메서드/ pop() 메서드

```
문자열 변수 = 배열.shift( )
```

- 배열의 첫 번째 요소를 제거하고 제거된 문자열을 반환

```
문자열변수 = 배열.pop( )
```

- 배열의 마지막 요소를 제거하고 제거된 문자열을 반환

chapter16 > pushUnshift.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <script>
6
7          let nums = [30, 40, 50];
8
9          document.write("원래 배열 : ", nums, "<br><br>");
10
11         nums.unshift(10, 20);
12         document.write("unshift 10, 20 : ", nums, "<br>");
13
14         nums.push(60, 70);
15         document.write("push 60, 70 : ", nums, "<br><br>");
16
17         let first = nums.shift();
18         document.write("shift : ", first, "<br>");
19
20         let last = nums.pop();
21         document.write("pop : ", last, "<br><br>");
22
23         document.write("현재 배열 : ", nums);
24
25     </script>
26 </head>
27 <body></body>
28 </html>
```

```
원래 배열 : 30,40,50

unshift 10, 20 : 10,20,30,40,50
push 60, 70 : 10,20,30,40,50,60,70

shift : 10
pop : 70

현재 배열 : 20,30,40,50,60
```

Array 내장 객체의 메서드

- **splice() 메서드**
  - 배열의 내부에 있는 요소를 추출하거나 삽입하기 위한 메서드

```
변수 = 배열.splice( s, m, data )
```

  - 인덱스가 s인 위치부터 m개의 요소를 제거한 다음 data를 삽입
  - 제거된 요소를 변수에 반환

  - 요소가 하나인 경우 s로 인식

```
변수 = 배열.splice( s )
```

    배열의 인덱스가 s인 위치부터 끝까지 제거
    제거된 요소를 반환

  - 요소가 두 개인 경우 s와 m으로 인식

```
변수 = 배열.splice( s, m )
```

    배열의 인덱스가 s인 위치부터 m개의 요소를 제거
    제거된 요소를 반환

chapter16 > splice.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        let arr1 = [10, 20, 30, 40, 50];

        document.write("원래 배열 : ", arr1, "<br><br>");

        let data1 = arr1.splice(2, 0, "a");
        document.write("변경된 배열 : ", arr1, "<br>");

        let data2 = arr1.splice(2, 2, "b");
        document.write("변경된 배열 : ", arr1, "<br>");
        document.write("반환값 : ", data2, "<br><br>");

        let data3 = arr1.splice(2, 1);
        document.write("변경된 배열 : ", arr1, "<br>");
        document.write("반환값 : ", data3, "<br><br>");

        let data4 = arr1.splice(2);
        document.write("변경된 배열 : ", arr1, "<br>");
        document.write("반환값 : ", data4, "<br><br>");

    </script>
</head>
<body></body>
</html>
```

```
원래 배열 : 10,20,30,40,50

변경된 배열 : 10,20,a,30,40,50
변경된 배열 : 10,20,b,40,50
반환값 : a,30

변경된 배열 : 10,20,40,50
반환값 : b

변경된 배열 : 10,20
반환값 : 40,50
```

Array 내장 객체의 메서드

배열 요소의 추출

- slice() 메서드
  - m 위치부터 n위치까지의 요소를 추출하여 새로운 배열 생성

```
배열2 = 배열1.slice( m, n )
```

  - 배열1의 m위치부터 n위치까지 추출하여 배열2를 생성
    - m 위치 : 0부터 시작
    - n 위치 : 1부터 시작
  - 인자를 하나만 지정할 경우 m으로 인식
    - 지정한 인덱스 m부터 마지막까지 추출

```
원래 배열 : 10,20,30,40,50

arr2 배열 : 30,40

arr2 배열 : 30,40,50

현재 배열 : 10,20,30,40,50
```

chapter16 > slice.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        let arr1 = [10, 20, 30, 40, 50];

        document.write("원래 배열 : ", arr1, "<br><br>");

        let arr2 = arr1.slice(2,4);
        document.write("arr2 배열 : ", arr2, "<br><br>");

        let arr3 = arr1.slice(2);
        document.write("arr2 배열 : ", arr3, "<br><br>");

        document.write("현재 배열 : ", arr1, "<br><br>");

    </script>
</head>
<body></body>
</html>

```

Array 내장 객체의 메서드

배열 요소의 정렬

- reverse() 메서드

```
배열.reverse( )
```

- 배열 요소의 순서를 역순으로 반환

  원래의 배열 내용이 변화됨 (정렬이 아님)

- 사용 예

```
배열1.reverse()
```

배열1의 내용을 역순으로 바꾼 후 다시 배열1에 저장

```
배열2 = 배열1.reverse()
```

배열1의 내용을 역순으로 바꾼 후 배열1과 배열2에 저장

배열1의 내용 = 배열2의 내용

chapter16 > reverse.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        let arr1 = ["홍길동", "이순신", "강감찬", "권율"];

        for (const i in arr1) {
            document.write("arr1["+i+"] : ", arr1[i], "<br>");
        }

        arr1.reverse();

        for (const i in arr1) {
            document.write("arr1["+i+"] : ", arr1[i], "<br>");
        }

    </script>
</head>
<body></body>
</html>

```

```
arr1[0] : 홍길동
arr1[1] : 이순신
arr1[2] : 강감찬
arr1[3] : 권율
arr1[0] : 권율
arr1[1] : 강감찬
arr1[2] : 이순신
arr1[3] : 홍길동
```

Array 내장 객체의 메서드

- **sort() 메서드**
  - 배열의 각 요소를 오름차순으로 정렬하여 반환

```
배열.reverse( )
```

숫자 정렬이 아닌 문자 정렬을 수행

원래의 배열을 정렬하여 저장 (원래의 배열 내용이 변화됨)

  - 사용 예

```
배열1.sort()
```

배열1의 내용을 오름차순 정렬한 후 다시 배열1에 저장

```
배열2 = 배열1.sort()
```

배열1의 내용을 오름차순으로 정렬하여 배열2에 저장

배열1도 정렬됨

  - 내림차순 정렬을 수행하는 메서드는 존재하지 않음

sort()와 reverse() 함께 사용

chapter16 > sort.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        let arr1 = ["홍길동", "이순신", "강감찬", "권율"];

        for (const i in arr1) {
            document.write("arr1["+i+"] : ", arr1[i], "<br>");
        }

        arr1.sort();

        for (const i in arr1) {
            document.write("arr1["+i+"] : ", arr1[i], "<br>");
        }

        arr1.sort().reverse();

        for (const i in arr1) {
            document.write("arr1["+i+"] : ", arr1[i], "<br>");
        }

    </script>
</head>
<body></body>
</html>
```

```
arr1[0] : 홍길동
arr1[1] : 이순신
arr1[2] : 강감찬
arr1[3] : 권율
arr1[0] : 강감찬
arr1[1] : 권율
arr1[2] : 이순신
arr1[3] : 홍길동
arr1[0] : 홍길동
arr1[1] : 이순신
arr1[2] : 권율
arr1[3] : 강감찬
```

Array 내장 객체의 메서드

- **숫자 정렬**

  - 오름차순 정렬

```
arr1.sort(function(left,right) {
        return left-right;
});
```

  - 내림차순 정렬

```
arr1.sort(function(left,right) {
        return right-left;
});
```

chapter16 > sort-number.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <script>
6
7           let arr1 = [22, 156, 77, 342, 16];
8
9           for (const i in arr1) {
10              document.write("arr1["+i+"] : ", arr1[i], "<br>");
11          }
12
13          arr1.sort(function(left,right) {
14              return left-right;
15          });
16          for (const i in arr1) {
17              document.write("arr1["+i+"] : ", arr1[i], "<br>");
18          }
19
20          arr1.sort(function(left,right) {
21              return right-left;
22          });
23          for (const i in arr1) {
24              document.write("arr1["+i+"] : ", arr1[i], "<br>");
25          }
26
27      </script>
28  </head>
29  <body></body>
30  </html>
```

```
arr1[0] : 22
arr1[1] : 156
arr1[2] : 77
arr1[3] : 342
arr1[4] : 16
arr1[0] : 16
arr1[1] : 22
arr1[2] : 77
arr1[3] : 156
arr1[4] : 342
arr1[0] : 342
arr1[1] : 156
arr1[2] : 77
arr1[3] : 22
arr1[4] : 16
```

Array 내장 객체의 메서드

배열 요소의 위치 추출

- indexOf() 메서드

```
변수 = 배열.indexOf( 인자 )
```

- 배열의 처음(왼쪽) 요소부터 조사

  인자로 지정된 요소가 존재하면 해당 index를 반환

  index의 기준은 맨 좌측 요소를 사용(index=0)

  존재하지 않으면 -1을 반환

- lastIndexOf() 메서드

```
변수 = 배열.lastIndexOf( 인자 )
```

- 배열의 마지막(오른쪽) 요소부터 조사

  인자로 지정된 요소가 존재하면 해당 index를 반환

  index의 기준은 맨 좌측 요소를 사용(index=0)

  존재하지 않으면 -1을 반환

chapter16 > indexOf.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <script>
6
7           const array = [1, 2, 3, 4, 5, 5, 4, 3, 2, 1];
8
9           const output1 = array.indexOf(4);
10          document.write("output1 : ", output1, "<br>");
11
12          const output2 = array.lastIndexOf(4);
13          document.write("output2 : ", output2, "<br>");
14
15          const output3 = array.indexOf(8);
16          document.write("output3 : ", output3, "<br>");
17
18      </script>
19  </head>
20  <body></body>
21  </html>
```

```
output1 : 3
output2 : 6
output3 : -1
```

Array 내장 객체의 메서드

배열 요소 관련 함수

- forEach() 메서드
  - 반복문을 사용해 배열의 각 요소에 대해 특정한 함수를 모두 적용

```
forEach ( function( element, index ) {

            적용할 코드

});
```

  - element
    현재 반복문에서 수행되는 배열 요소의 값
  - index
    현재 반복문에서 수행되는 배열 요소의 index

chapter16 > forEach.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        const array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

        let sum = 0;
        let output = '';

        array.forEach(function (element, index) {
            sum = element * 10;
            output += index + ": " + sum + "<br>";
        });

        document.write(output);

    </script>
</head>
<body></body>
</html>
```

```
0: 10
1: 20
2: 30
3: 40
4: 50
5: 60
6: 70
7: 80
8: 90
9: 100
```

Array 내장 객체의 메서드

- **every() 메서드**
  - 배열의 모든 요소가 특정 조건을 만족하는지 조사
  - 모두 만족할 경우 true를 반환
    하나의 요소라도 조건을 만족하지 않으면 false를 반환

```
변수 = 배열.every( function( element ) {
            지정할 조건
} );
```

- **some() 메서드**
  - 배열의 모든 요소 중 하나 이상이 특정 조건을 만족하는지 조사
  - 하나 이상 만족할 경우 true를 반환

```
변수 = 배열.some( function( element ) {
            지정할 조건
} );
```

- **filter() 메서드**
  - 기존 배열 요소 중에서 특정 조건을 만족하는 요소만을 추출해 새로운 배열을 생성

```
변수 = 배열.filter( function( element ) {
            지정할 조건
} );
```

  - element
    배열 내의 각 요소들이 순차적으로 입력되는 인자

Array 내장 객체의 메서드

chapter16 > every-some.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        var myArray = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

        function lessThenFive(element) {
            return element < 5;
        }

        var result1 = myArray.every(lessThenFive);
        document.write(result1, "<br><br>");

        var result2 = myArray.some(lessThenFive);
        document.write(result2, "<br><br>");

        var result3 = myArray.filter(lessThenFive);
        document.write(result3);

    </script>
</head>
<body></body>
</html>
```

false

true

1,2,3,4

Date 객체

- **날짜와 시간 관련 정보를 제공하는 내장 객체**
  - 반드시 new 키워드를 사용해 객체(인스턴스)를 생성해 사용해야 함

```
인스턴스 =  new Date( [시간정보] )
```

- **시간 객체 생성을 통한 시간 정보 추출**
  - 현재 시간 정보를 가진 객체 생성 가능
    현재의 년, 월, 일, 시, 분, 초 정보 추출 가능
  - 임의의 날짜에 해당하는 객체 생성 가능
    임의의 날짜에 대한 년, 월, 일, 시, 분, 초, 요일 정보 추출 가능

- **시간 객체의 정보 변경**
  - 기존의 객체가 가지는 시간 정보 변경 가능
    요일 정보는 변경 불가능

Date 객체

Date 객체의 생성자

- 현재 시간정보를 가진 Date 객체 생성

```
인스턴스 = new Date();
```

```
let day1 = new Date();
```

- 지정한 시간 정보를 가진 Date 객체 생성

```
인스턴스 = new Date("YYYY-(MM-1)-DD");
인스턴스 = new Date("YYYY/MM-1/DD");
인스턴스 = new Date("MM-1 DD YYYY");
인스턴스 = new Date("YYYY MM-1 DD");
인스턴스 = new Date("DD/MM-1/YYYY");
인스턴스 = new Date("YYYY-(MM-1)-DDTHH:MM:SS")
인스턴스 = new Date("YYYY/MM-1/DD HH:MM:SS");
```

```
let day1 = new Date("2023-11-25");          // 2023년 12월 25일
let day2 = new Date("2023/11/25");          // 2023년 12월 25일
let day3 = new Date("11 25 2023");          // 2023년 12월 25일
let day4 = new Date("2023 11 25");          // 2023년 12월 25일
let day5 = new Date("11/25/2023");          // 2023년 12월 25일
let day6 = new Date("2023-11-25T14:30:25")  // 2023년 12월 25일 14시 30분 25초
let day7 = new Date("2023/11/25 14:30:25"); // 2023년 12월 25일 14시 30분 25초
```

Date 객체의 메서드

Date 객체의 메서드

- 시간정보 추출

메서드	설명
getFullYear()	년도 정보를 출력
getMonth()	월 정보를 출력(0~11) [1월(0) ~ 12월(11)]
getDate()	일 정보를 출력(1~31)
getDay()	요일 정보를 출력(0~6) [일요일 : 0, 토요일 : 6]
getHours()	시간 정보를 출력(0~23)
getMinutes()	분 정보를 출력(0~59)
getSeconds()	초 정보를 출력(0~59)
getMiliseconds()	밀리 초(1/1000 초) 정보를 출력
getTime()	1970년 1월 1일부터 지정한 날짜 정보까지를 1/1000초로 표시
toGMTString()	GMT 표준 표기 방식으로 문자열 데이터로 변환해 출력
toLocaleString()	로컬 시간대 표기 방식으로 문자열 데이터로 변환해 출력

Date 객체의 메서드

- 시간 정보 지정

메서드	설명
setFullYear()	년도 정보를 지정
setMonth()	월 정보를 지정(1월 : 0, 12월 : 11)
setDate()	일 정보를 지정(1~31)
setHours()	시간 정보를 지정(0~23)
setMinutes()	분 정보를 지정(0~59)
setSeconds()	초 정보를 지정(0~59)
setMiliseconds()	밀리 초(1/1000 초) 정보를 지정

Date 객체의 메서드

chapter16 > Date1.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>
        let td = new Date( );

        document.write("<h3>오늘 날짜 정보</h3>");
        document.write("오늘은 ");
        document.write(td.getFullYear(),"년 ", td.getMonth()+1,"월 ", td.getDate());
        document.write("일입니다<br>");

        day = td.getDay();
        switch(day) {
            case(0) : now_day = "일요일"; break;
            case(1) : now_day = "월요일"; break;
            case(2) : now_day = "화요일"; break;
            case(3) : now_day = "수요일"; break;
            case(4) : now_day = "목요일"; break;
            case(5) : now_day = "금요일"; break;
            case(6) : now_day = "토요일"; break;
        }
        document.write("오늘은 ",now_day,"입니다<br>");
        document.write("지금은 ");
        document.write(td.getHours(),"시 ",td.getMinutes(),"분 ",td.getSeconds(),"초입니다");
    </script>
</head>
<body></body>
</html>
```

오늘 날짜 정보

오늘은 2023년 2월 7일입니다
오늘은 화요일입니다
지금은 11시 21분 32초입니다

Date 객체의 메서드

chapter16 > Date2.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        const today = new Date();
        const nowYear = today.getFullYear();

        const theDate = new Date(nowYear, 11, 31);
        const diffDate = theDate.getTime( ) - today.getTime( );

        const result = Math.ceil( diffDate / (60 * 1000 * 60 * 24) );

        document.write("오늘은 ", today.getMonth()+1,"월 ", today.getDate(), "일입니다<br>" );
        document.write("<h3>오늘부터 올해 연말까지" ,result,"일 남았습니다</h3>");

    </script>
</head>
<body></body>
</html>
```

오늘은 2월 7일입니다

오늘부터 올해 연말까지327일 남았습니다

Math 객체

Math 객체

- **각종 산술 연산에 활용될 수 있는 속성과 메소드를 포함하고 있는 객체**
  - 삼각함수, 지수함수, 로그함수, 각종 상수, 제곱근 등

- **Math 객체는 정적 객체임**
  - 인스턴스를 생성하지 않고 객체 자신이 직접 인스턴스로 사용되는 객체
  - new 연산자로 생성 불가능

    [예] m = new Math(); (불가능)
  - 객체 자신이 직접 메소드와 속성을 호출해 사용

    [예] Math.sin();

    [예] Math.Pi;

Math 객체의 속성과 메서드

- Math 객체의 자주 사용되는 속성과 메서드

속성	설명
E	오일러 상수
PI	원주율
LN2	$log_e 2$
LN10	$log_e 10$

속성	설명
abs(값)	인자로 가지는 값의 절대값을 반환
max(값 리스트)	인자로 가지는 값 리스트 중 가장 큰 값을 반환
min(값 리스트)	인자로 가지는 값 리스트 중 가장 작은 값을 반환
power(x, y)	인자로 가지는 x값의 y승 값을 반환
random()	0에서 1까지의 난수 값을 반환
round(값)	인자로 가지는 값을 소수점 첫째 자리에서 반올림하여 반환
ceil(값)	인자로 가지는 값보다 같거나 작은 수 중에서 가장 큰 정수 값을 반환
floor(값)	인자로 가지는 값보다 같거나 큰 수 중에서 작은 큰 정수 값을 반환
sqrt(값)	인자로 가지는 값의 제곱근을 반환
sin()	사인(sine) 값을 반환
cos()	코사인(cosim) 값을 반환
tan()	탄젠트(tangent) 값을 반환

Math 객체의 속성과 메서드

chapter16 > math1.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        const num = 2.1234;
        const maxNum = Math.max(10, 5, 8, 30);
        const minNum = Math.min(10, 5, 8, 30);

        document.write("max(10, 5, 8, 30) : ", maxNum, "<br>");
        document.write("min(10, 5, 8, 30) : ", minNum, "<br>");

        document.write("round(2.1234) : ", Math.round(num), "<br>");
        document.write("flooe(2.1234) : ", Math.floor(num), "<br>");
        document.write("ceil(2.1234) : ", Math.ceil(num), "<br>");
        document.write("random : ", Math.random(), "<br>");

        document.write("PI : ", Math.PI, "<br>");

    </script>
</head>
<body></body>
</html>
```

```
max(10, 5, 8, 30) : 30
min(10, 5, 8, 30) : 5
round(2.1234) : 2
flooe(2.1234) : 2
ceil(2.1234) : 3
random : 0.16410384458118288
PI : 3.141592653589793
```

Math 객체의 속성과 메서드

chapter16 > math2.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        document.write("<h3>10부터 50 사이의 난수</h3>")

        for (let i=0; i<10; i++) {
            let num =  Math.floor(Math.random()*(50-10+1)+10);
            document.write("10에서 50까지의 난수 ", i , " : ", num ,"<br>");
        }

    </script>
</head>
<body></body>
</html>
```

10부터 50 사이의 난수

10에서 50까지의 난수 0 : 31
10에서 50까지의 난수 1 : 24
10에서 50까지의 난수 2 : 47
10에서 50까지의 난수 3 : 15
10에서 50까지의 난수 4 : 12
10에서 50까지의 난수 5 : 18
10에서 50까지의 난수 6 : 11
10에서 50까지의 난수 7 : 46
10에서 50까지의 난수 8 : 41
10에서 50까지의 난수 9 : 37

String 객체

- **문자열 연산을 수행하고 문자열로부터 정보를 추출하는 객체**
- **문자열 객체를 생성하는 방법**
  - new 연산자와 String() 객체를 사용하는 방법

```
변수 = new String("문자열")
```

```
let str = new String("korea");
```

  - 문자 리터럴을 직접 지정하여 생성하는 방법

```
변수 = 문자열
```

```
let  str = "korea";
```

- **String 객체의 속성**
  - 문자열의 길이를 반환하는 length라는 유일한 속성을 제공

chapter16 > String-length.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <script>
6
7           const str1 = new String("korea");
8           document.write("str1의 길이 : ", str1.length, "<br>");
9
10          const str2 = "대한민국";
11          document.write("str2의 길이 : ", str2.length, "<br>");
12
13      </script>
14  </head>
15  <body>
16
17  </body>
18  </html>
```

```
str1의 길이 : 5
str2의 길이 : 4
```

String 객체의 메서드

문자의 위치 관련 메서드

- indexOf() 메서드/ lastIndexOf() 메서드

```
변수 = 문자열.indexOf("문자", [시작위치])
```

- 왼쪽부터 검색하여 인자로 가지는 문자의 위치 값을 숫자로 반환
  문자열 객체의 가장 왼쪽 문자의 위치 값은 0임

```
변수 = 문자열.lastIndexOf("문자", [시작위치])
```

- 오른쪽부터 검색하여 인자로 가지는 문자의 위치 값을 숫자로 반환
  위치 값의 기준은 문자열 객체의 왼쪽문자 (위치 값 : 0) 임
- [시작 위치]를 지정한 경우 지정된 index의 문자부터 검색

- charAt(index) 메서드
  - 문자열에서 인자로 지정한 index에 해당하는 위치의 문자를 반환

```
변수 = 문자열.charAt( index )
```

chapter16 > indexOfLastIndexof.htm > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <script>
6
7      const str = "i love korea";
8
9      //charAt()메서드
10     const data = str.charAt(7);
11     document.write("charAt(7) : ", data, "<br><br>");
12
13     //indexOf(), lastIndexOf() 메서드
14     const str1 = str.indexOf('o');
15     document.write("indexOf('o') : ", str1, "<br><br>");
16
17     const str2 = str.indexOf('o',5);
18     document.write("indexOf('o',5) : ", str2, "<br><br>");
19
20     const str3 = str.lastIndexOf('o');
21     document.write("lastIndexOf('o') : ", str3);
22
23     </script>
24 </head>
25 <body></body>
26 </html>
```

```
charAt(7) : k

indexOf('o') : 3

indexOf('o',5) : 8

lastIndexOf('o') : 8
```

String 객체의 메서드

문자 검색 관련 메서드

- search() 메서드

```
변수 = 문자열.search("문자")
```

- 왼쪽부터 검색해 인자로 지정한 문자가 나타나는 index를 반환
  발견되지 않을 경우 -1을 반환

- replace() 메서드

```
변수 = 문자열.replace( str1, str2 )
```

- 문자열을 검색하여 str1이 존재하는 경우 str2로 치환하여 반환
  str1이 존재하지 않을 경우 원래 문자열 전체를 반환

chapter16 > search-replace.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
</head>
<body>
    <script>

        const str = "i love korea";

        //search() 메서드
        document.write("love 위치 : ", str.search('love'), "<br>");
        document.write("like 위치 : ", str.search('like'), "<br>");

        //replace() 메서드
        const str1 = str.replace('love', 'like');
        document.write("love를 like로 : ", str1, "<br>");

        const str2 = str.replace('LOVE', 'like');
        document.write("LOVE를 love로 : ", str2, "<br>");

    </script>
</body>
</html>
```

```
love 위치 : 2
like 위치 : -1
love를 like로 : i like korea
LOVE를 love로 : i love korea
```

String 객체의 메서드

문자 추출 메서드

- substring() 메서드/ slice()메서드

```
변수 = 문자열.substring( m, n )
```

```
변수 = 문자열.slice( m, n )
```

- index 값이 m인 위치 문자부터 n인 위치 문자까지 추출
  - 추출한 값으로 새로운 문자열을 반환
  - m의 경우 첫번째 문자의 index는 0임
  - n의 경우 첫번째 문자의 index는 1임

- substr() 메서드
  - index 값이 m인 위치부터 n개의 문자열을 추출해 반환

```
변수 = 문자열.substr( m, n )
```

chapter16 > substr-slice.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5  </head>
6  <body>
7      <script>
8
9          const str = "more than words";
10
11         // substring() 메서드
12         const str1 = str.substring(5,9);
13         document.write("substring(5,9) : ", str1, "<br><br>");
14
15         // slice() 메서드
16         const str2 = str.slice(5,9);
17         document.write("slice(5,9) : ", str2, "<br><br>");
18
19         // substr() 메서드
20         str3 = str.substr(5,4);
21         document.write("substr(5,4) : ", str3);
22
23     </script>
24 </body>
25 </html>
```

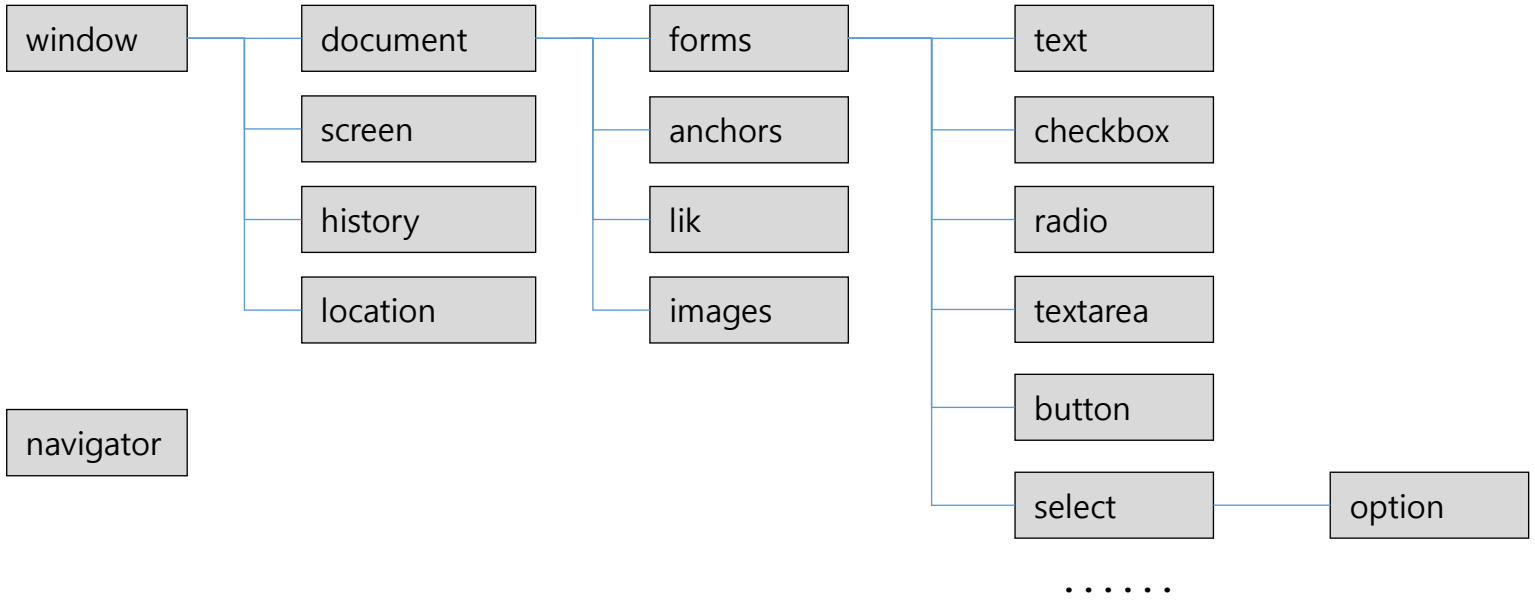
```
substring(5,9) : than

slice(5,9) : than

substr(5,4) : than
```

16-3. 브라우저와 관련된 객체들

브라우저와 관련된 객체

브라우저가 제공하는 객체

- **자바스크립트에서 사용 가능**
  - 실제 자바스크립트에서는 자바스크립트 내장 객체보다 브라우저 객체가 더 많이 사용됨
  - 크게 Window 객체와 Navigator 객체로 구분



브라우저와 관련된 객체

window 객체와 navigator 객체

- Window 객체
  - 브라우저 최상위 객체로 하위에 많은 객체를 포함하고 있음
    document 객체, screen 객체, location 객체, history 객체
  - window 객체의 하위 객체
    document 객체 : 웹 문서와 그 내용에 대한 처리를 위한 객체 (가장 큰 객체)
    screen 객체 : 모니터와 출력 영역의 정보를 추출하기 위한 객체
    location 객체 : URL을 처리하기 위한 객체
    history 객체 : 히스토리 정보를 유지하고 처리하기 위한 객체

- Navigator 객체
  - 브라우저의 정보를 유지하는 객체

window 객체

window 객체의 주요 메서드

메서드	설명
open()/ close()	새로운 윈도를 생성/ 생성된 윈도우를 닫음
alert()	특정 문자열과 [확인] 버튼을 가진 대화상자 출력
prompt()	특정 문자열과 데이터 입력창, [확인]과 [취소] 버튼을 가진 대화상자 출력
confirm()	특정 문자열과 [확인]과 [취소] 버튼을 가진 대화상자 출력
moveTo()/ moveBy()	윈도우를 절대적(moveTo()) 또는 상대적인 위치 (moveBy())로 이동
resizeTo()/ resizeBy()	윈도우를 절대적(resizeTo()) 또는 상대적인 크기 (resizeBy())로 재지정
setIntervel()	일정 간격으로 특정 명령을 반복 수행
clearInterval()	setIntervel()을 사용해 지정된 반복 작업을 해제
setTimeout()	일정 시간 후 특정 명령을 수행
clearTimeOut()	setTimeout()을 사용해 지정된 작업을 해제

윈도우의 생성과 종료

- open() 메서드
  - 새로운 윈도우를 생성하는 메서드

```
open( "열릴 문서",  "윈도우 이름",  "윈도우 속성" )
```

```
윈도우 이름 = open( "열릴 문서", "윈도우` 속성" )
```

  - 열릴 문서
    생성되는 윈도우에 포함될 웹 문서의 이름이나 사이트의 주소
  - 윈도우 이름
    생성되는 고유한 윈도우 이름
    여러 윈도우를 식별하거나 윈도우를 제어하기 위해 사용
  - 윈도우 속성
    생성되는 윈도우가 가지는 속성을 지정
    윈도우를 구성하는 요소의 출력 여부를 yes나 no로 지정
    윈도우의 위치나 크기 등을 픽셀 단위로 지정

- 윈도우 속성 항목

항목	값	설명
width/height	pixel	윈도우의 너비와 높이지정
left/top	pixel	생성되는 윈도우의 좌측 상단 좌측/우측 좌표 지정
location	yes/no	주소 입력 줄 표시 여부 지정 (기본값 : no)
status	yes/no	상태표시줄 표시 여부 지정 (기본값 : no)
scrollbars	yes/no	스크롤 바 표시 여부 지정 (기본값 : no) (브라우저 버전에 따라 출력되지 않을 수 있음)
toolbar	yes/no	툴 바의 표시 여부 지정 (기본값 : no)
resizable	yes/no	윈도우 크기 조절 가능 여부 지정 (기본값 : no)

window 객체

- close() 메서드
  - 생성되어 있는 윈도우를 닫기 위한 메서드

```
윈도우이름.close()
```

  - 다른 윈도우를 닫을 수 있으며, 자기 자신의 윈도우 닫을 수도 있음

[예] 자신의 윈도우를 닫을 경우

```
window.close()  또는  self.close()
```

[예] login이라는 이름의 윈도우를 닫을 경우

```
login.close()
```

window 객체

chapter16 > window-open-close.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        let win2 = null

        function win_open() {
            win2 = window.open('https://www.ut.ac.kr','','width=1000,height=500,left=200,top=200');
        }

    </script>
</head>
<body>

    <input type=button value="open" onClick="win_open()">
    <input type=button value="close" onClick="win2.close()">

</body>
</html>
```

window 객체

chapter16 > popup-check.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        let blocked = false;

        function openPopup() {
            var newWin = window.open("notice.html","pop","width=500,height=400");
            if (newWin == null) {
                alert("팝업이 차단되어 있습니다. 팝업 차단을 해제해 주세요.")
            }
            newWin.moveBy(200,200);
        }

    </script>
</head>
<body onload="openPopup()">
    <p>문서를 열면 팝업 창이 표시됩니다</p>
</body>
</html>
```

chapter16 > notice.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <style>
        #content {
          border : 2px double skyblue;
          border-radius:10px; padding:10px;
        }
        ul { margin-left:15px; list-style-type:none; }
        ul li {margin : 10px 5px;}
      </style>
</head>
<body>
    <div id="content">
        <h1>공지사항</h1>
          <ul>
            <li>항목 1</li>
            <li>항목 2</li>
            <li>항목 3</li>
            <li>항목 4</li>
          </ul>
        </div>
          <button onclick="javascript:window.close();">닫기</button>
</body>
</html>
```

일정한 시간 후 명령 실행

- **setTimeout() 메서드**
  - 인자로 가지는 실행 문장을 지정된 시간 후에 한번 실행함 (예약 실행의 개념)

```
시간 아이디 = setTimeout( 실행문장,  시간 )
```

  - 시간은 1/1,000ch를 사용 (1초일 경우 1,000을 지정)

    [예] now = setTimeout("check()", 1000)

- **clearTimeout() 메서드**
  - setTimeout() 함수에 의해 수행되는 실행 문장이 실행을 중지시킴

```
clearTimeout( 시간 아이디 )
```

  - 중지시킬 실행 문장은 setTimeout() 함수에서 지정한 시간 ID를 사용

    [예] clearTimeout(now)

  - 한 번만 수행되므로 setTimeout()에 의해 수행되기 전에 취소할 경우 사용

window 객체

chapter16 > setTimeout.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        function start() {
            time_id = setTimeout("alert('Hello')", 2000);
        }

        function stop() {
            clearTimeout(time_id);
        }

      </script>
</head>
<body>
    <div id="content">

    <input type=button  onClick="start()" value="시작">
    <input type=button  onClick="stop()" value="종료" >

</body>
</html>
```

명령의 반복 수행

- **setInterval() 메서드**
  - 인자로 가지는 실행 문장을 지정된 시간마다 반복하여 실행함 (순환의 개념)

```
시간 아이디 = setInterval( 실행문장,  시간 )
```

  - 시간은 1/1,000ch를 사용 (1초일 경우 1,000을 지정)

    [예] now = setInterval("check()", 1000

- **clearInterval() 메서드**
  - setInterval() 함수에 의해 수행되는 실행문장의 실행을 중지시킴

```
clearInterval( 시간 아이디 )
```

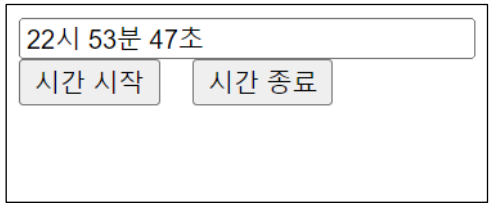
  - 중지시킬 실행 문장은 setInterval() 함수에서 지정한 시간 ID를 사용

    [예] clearInterval(now)

window 객체

chapter16 > setInterval.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        let time_id = null;

        function time_start() {
            today = new Date();
            now = today.getHours()+"시 " + today.getMinutes()+"분 " + today.getSeconds()+"초";
            document.myform.timebar.value=now;
        }
        function start() {
            time_id = setInterval("time_start()", 1000);
        }
        function stop() {
            clearInterval(time_id);
            document.myform.timebar.value=null
        }

    </script>
</head>
<body>
    <form name="myform">
        <input type=text name=timebar SIZE=30><br>
        <input type=button value="시간 시작" onClick="start()">  
        <input type=button VALUE="시간 종료" onClick="stop()">
    </form>
</body>
</html>
```



screen 객체

screen 객체

- 사용자의 모니터 정보를 제공하는 객체
  - 속성은 read-only임 (속성값은 수정이 불가능함)
  - 메서드는 존재하지 않음

- screen 객체의 속성

속성	설명
width	전체 화면의 너비 정보를 출력
height	전체 화면의 높이 정보를 출력
availWidth	작업표시줄을 제외한 화면의 너비 정보를 출력
availHeight	작업표시줄을 제외한 화면의 높이 정보를 출력
colorDepth	현재의 모니터가 표현 가능한 컬러의 비트 수를 반환

chapter16 > screen.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <script>
6           document.write("width: ", screen.width, "<br>");
7           document.write("height: ", screen.height, "<br>");
8           document.write("availWidth: ", screen.availWidth, "<br>");
9           document.write("availHeight: ", screen.availHeight, "<br>");
10          document.write("colorDepth: ", screen.colorDepth);
11      </script>
12  </head>
13  <body>
14
15  </body>
16  </html>
```

```
width: 1707
height: 1067
availWidth: 1707
availHeight: 1036
colorDepth: 24
```

location 객체

location 객체

- 주소 표시줄에 입력된 주소 정보를 추출하거나 다른 문서나 사이트로의 이동을 위한 객체

- 메서드
  - 다른 문서나 사이트로 이동하기 위해 사용

- 속성
  - 웹 브라우저의 주소표시줄에 입력된 주소에 대한 정보를 추출
  - 주소 표시줄의 프로토콜, 호스트 이름, 문서의 위치 등의 정보를 추출할 수 있음
  - 실제 인터넷을 통해 수행되는 경우에만 모든 정보 확인 가능

- href 속성

```
location.href='이동할 문서(사이트)'
```

  - 지정한 문서(사이트)로 이동

- reload() 메서드
  - 현재 페이지에 대해 [새로 고침]을 수행하는 메서드

```
location.reload()
```

    브라우저에서 reload 기능과 동일한 역할 수행

- replace() 메서드
  - 인자로 지정한 문서나 사이트로 이동하기 위한 메서드

```
location.replace('이동할 문서(사이트)')
```

    브라우저에서 [뒤로 가기] 버튼을 활성화하지 않음

- assign() 메서드
  - 인자로 지정한 문서나 사이트로 이동하기 위한 메서드

```
location.assign('이동할 문서(사이트)')
```

location 객체

chapter16 > location.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
</head>
<body>

    <input type=button value="새로고침" onClick="location.reload()"><br>

    <input type=button value="한국교통대학교" onClick="location.replace('https://www.ut.ac.kr')"><br>

    <input type=button value="네이버" onClick="location.assign('https://www.naver.net')"><br>

    <input type=button value="구글" onClick="location.href='https://www.google.com'"><br>

</body>
</html>
```

location 객체

- location 객체의 속성

속성	설명
hash	URL에 표시된 해시정보(서버이름 다음의 #이후 문자열)을 반환
hostname	URL에 표시된 호스트의 이름을 반환
host	URL에 표시된 호스트의 이름과 포트 번호를 반환
port	URL에 표시된 포트 번호를 반환
pathname	URL에 표시된 서버 이하의 디렉터리 경로를 반환
protocol	URL에 표시된 프로토콜을 반환
search	URL에 표시된 쿼리 스트링(?이하의 문자)을 반환
href	URL의 전체 정보를 반환
href=URL	인자로 지정된 URL로 이동함 [assign( ) 메서드와 동일한 기능 수행]

history 객체

- 사용자가 방문한 사이트의 히스토리 정보를 이용해 이동을 수행하기 위한 객체
- history객체의 속성과 메서드

구분		설명
속성	history.length	저장되어 있는 히스토리 정보의 수를 반환
메서드	history.back()	히스토리 정보 중 현재 위치를 기준으로 한 단계 이전 페이지로 이동
	history.forward()	히스토리 정보 중 현재 위치를 기준으로 한 단계 다음 페이지로 이동
	history.go(n)	히스토리 정보 중 현재 위치를 기준으로 인자로 가지는 n단계 만큼 이동 - n이 양수일 경우 : n단계 만큼 다음 페이지 출력 - n이 음수일 경우 : n단계 만큼 이전 페이지 출력 - history.go(1) = history.forward() - history.go(-1) = history.back() - history.go(0) = location.reload() // 새로 고침

document 객체

document 객체

- window 객체의 하위 객체로 웹 문서에 대한 정보와 문서에 포함된 객체의 정보를 표현
- 하위에 많은 객체를 포함하고 있음
  - HTML 문서 내에 나타난 대부분의 구성요소를 객체로 표현됨
  - 주로 이미지, URL 정보, 하이퍼링크, 〈FORM〉 객체 등이 가장 사용됨
- 문서의 색상을 지정하기 위한 속성

속성	설명
bgColor	문서의 배경색을 지정
fgColor	문서에 존재하는 글자 색을 지정
linkColor = '색상'	문서에 존재하는 기본 하이퍼링크 색을 지정
alinkColor = '색상'	문서에 존재하는 현재 활성화된 하이퍼링크 색을 지정
vlinkColor = '색상'	문서에 존재하는 한번 이상 방문한 하이퍼링크 색을 지정

document 객체

chapter16 > <> document-color.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>
        function chgcolor() {
            document.bgColor = "black";
            document.fgColor = "gray";
            document.linkColor = "red";
            document.alinkColor = "green";
            document.vlinkColor = "blue";
        }
    </script>
</head>
<body onLoad="chgcolor()">

    <a href="#" target="win">MBC</a>사이트로 이동<P>
    <a href="#" target="win">KBS</a>사이트로 이동<P>
    <a href="#" target="win">SBS</a>사이트로 이동<P>

</body>
</html>
```

forms 객체

- **〈form〉 태그와 그 구성요소(element)들의 정보를 추출하고 관리하는 객체**
  - 자바스크립트에서의 forms 객체는 많은 하위 객체를 포함하고 있음
- **〈form〉 객체가 가지는 하위 객체**
  - text 객체, password 객체, textarea 객체, checkbox 객체, radio 객체, select 객체 등
  - 각각의 하위 객체들은 유사하거나 고유한 속성과 메소드를 가짐

forms 객체

〈form〉 요소와 하위 구성요소의 배열 표현

- **〈form〉 요소의 배열 표현**
  - 현재 문서에 하나 이상의 〈form〉 태그가 존재하는 경우, 모든 〈form〉 태그를 forms라는 배열로 반환

```
document.forms[ ]
```

  - [예] document.forms[0]
    현재 문서에 존재하는 첫 번째 <form> 태그를 의미

- **〈form〉 하위 요소의 배열 표현**
  - 〈form〉 태그에 존재하는 모든 구성요소를 나타나는 순서에 따라 element라는 배열로 반환

```
document.elements[ ]
```

  - [예] document.myform.element[0].value
    현재 문서에 존재하는 myform이라는 이름의 <form> 요소의 첫 번째 하위 요소에 입력된 값

forms 객체

- forms 객체의 속성과 메서드

속성과 메서드		설명
속성	name	<form> 태그에서 지정한 name 속성의 값을 반환
	action	<form> 태그에서 지정한 action 정보를 반환
	method	<form> 태그에서 지정한 method 정보를 반환 (post 또는 get)
	encoding	<form> 태그에서 지정한 인코딩 방식을 반환 - POST 방식의 데이터는 인코딩되어 전송됨 - 기본 인코딩 방식은 application/x-www-form-urlencoded임
	length	<form> 태그 내에 존재하는 구성 요소들의 수를 반환
메서드	submit( )	<form> 태그 내의 모든 입력 요소에 입력된 값을 action 속성이 지정하는 문서로 전송
	reset( )	<form> 태그 내의 모든 입력 요소에 입력된 값을 초기화

forms 객체

chapter16 > forms-text.html > ...

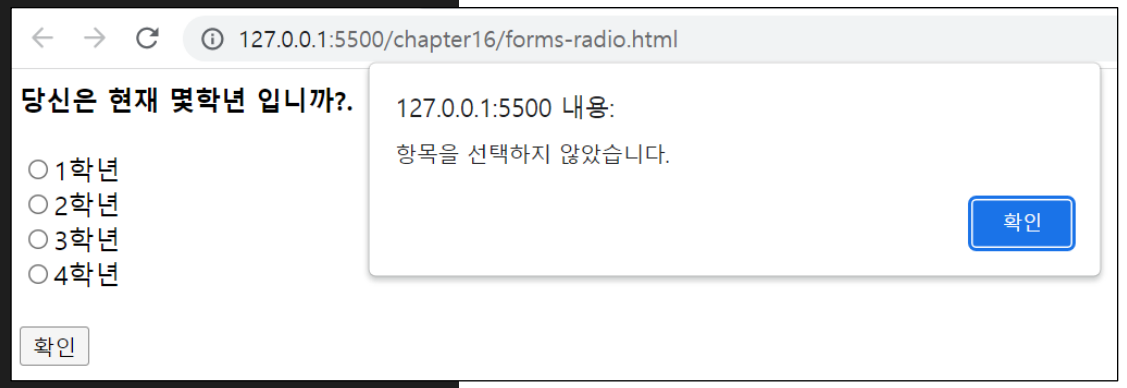
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>

        function sum(form) {
            const number1 = Number(form.num1.value);
            const number2 = Number(form.num2.value);
            const number3 = number1 + number2;
            form.num3.value = Number(number3);
        }

    </script>
</head>
<body onLoad="chgcolor()">

    <form name="myform">
        <input type=text name="num1" size=3> +
        <input type=text name="num2" size=3> =
        <input type=text name="num3" size=5> 
        <input type=button value="계산" onClick="javascript:sum(this.form)">
    </form>

</body>
</html>
```

30 + 70 = 100 계산

forms 객체

chapter16 > forms-radio.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script>
        function check(form) {
            const grade_len = form.grade.length;

            for(let i=0; i<grade_len; i++) {
                if(form.grade[i].checked==true) {
                    break;
                }
                if( i == grade_len-1 ) {
                    alert('항목을 선택하지 않았습니다.');
                }
            }

        }
    </script>
</head>
<body>
    <form name="myform">
        <b>당신은 현재 몇학년 입니까?.</b><br> <br>
        <input type=radio name="grade" value="1학년">1학년<br>
        <input type=radio name="grade" value="2학년">2학년<br>
        <input type=radio name="grade" value="3학년">3학년<br>
        <input type=radio name="grade" value="4학년">4학년<br> <br>
        <input type=button value="확인" onClick="javascript:check(this.form)">
    </form>
</body>
</html>
```



수고하셨습니다