객체

- 객체 지향 프로그래밍(OOP: object-oriented programming)은 우리가 사는 실제 세계가 객체(object)들로 구성된 것과 비슷하게, 소프트웨어도 객체로 구성하는 방법이다.

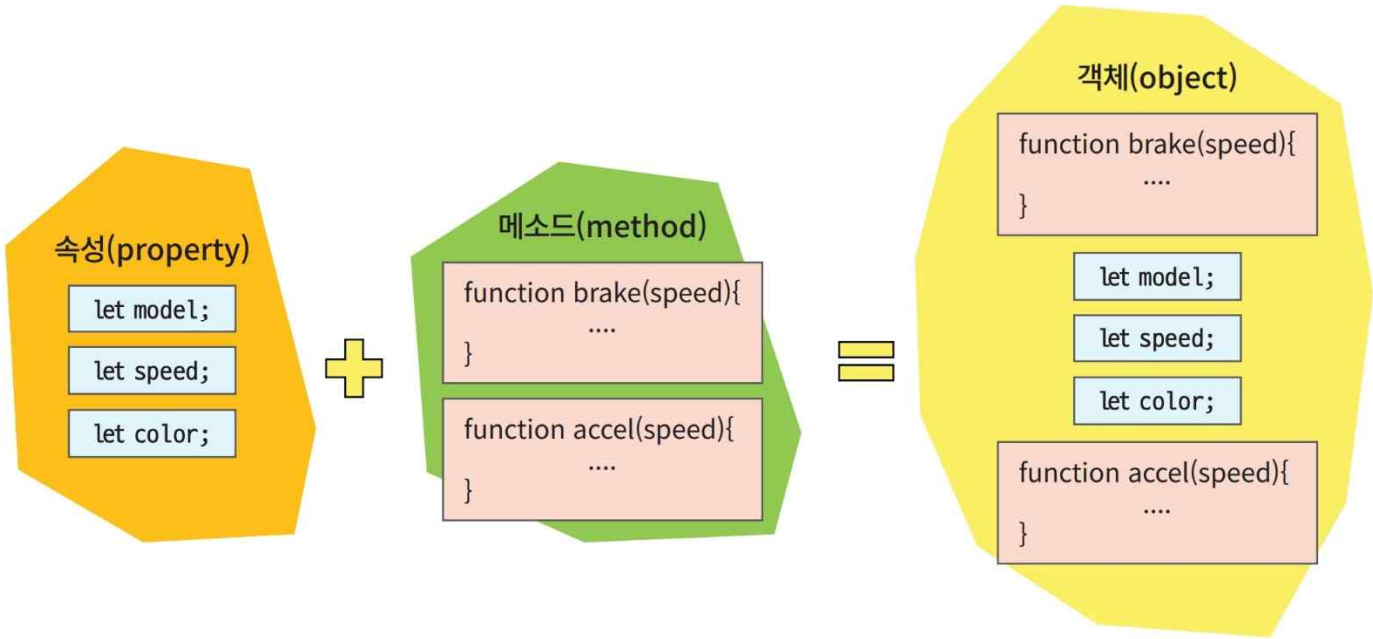
그림 8.1 객체들은 메시지를 주고받는다.

객체

- 객체는 데이터와 동작을 가지고 있다. 객체의 데이터(data)은 객체가 가지고 있는 특성값이다. 객체의 동작(action)은 객체가 수행할 수 있는 행동이다.

속성과 메소드

- 객체 안의 변수를 속성(property)라고 하고, 객체 안의 함수를 메소드(method)라고 부른다.



자바스크립트에서의 객체

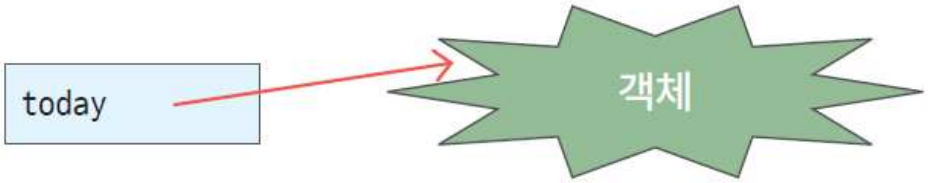
- 자바스크립트에서 사용되는 객체를 크게 2가지로 나누면 내장 객체와 사용자 정의 객체로 나눌 수 있다.
  - 내장 객체(built-in object): 자바스크립트에 내장된 객체이다. Date, String, Array와 같은 객체들이 내장 객체이다.
  - 사용자 정의 객체(custom object): 개발자가 정의하여 사용하는 객체이다.

- 내장 객체는 다시 핵심 객체와 DOM, BOM 객체로 나눌 수도 있다.
  - 핵심 객체: 기본적인 객체인 Date, String, Array들이 핵심 객체이다.
  - DOM(Document Object Model): 브라우저가 HTML 문서를 파싱하여서 각 요소들을 객체 트리로 정의한 것이다.
  - BOM(Brower Object Model): 브라우저의 종류나 크기 등의 정보를 제공하는 객체들이다.

객체 생성

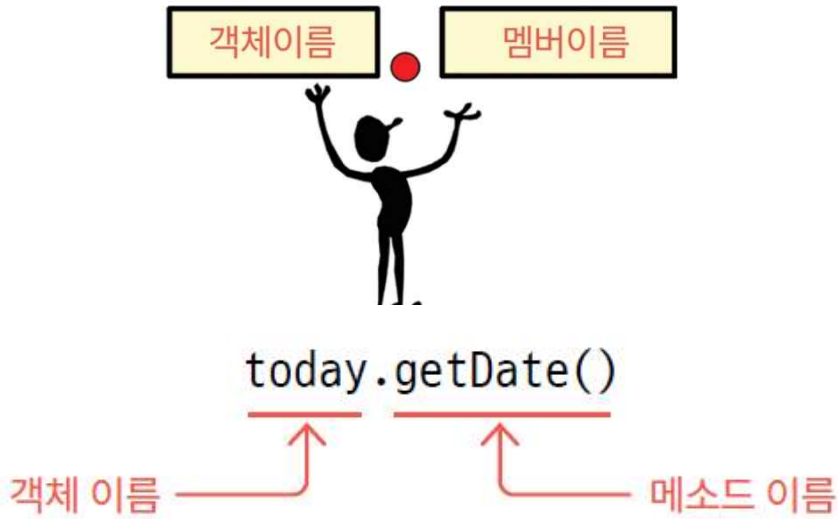
- 객체를 생성하기 위해서는 new 키워드를 사용한다.

```
let today = new Date();                    // 현재 날짜가 저장된 Date 객체가 생성된다.

let books = new Array("HTML", "Java", "C++");     // 배열이 생성된다.
```



객체 멤버 사용하기

- 객체 멤버들을 사용하기 위해서는 점(dot) 연산자를 사용한다. 먼저 어떤 객체에 속해 있는 지를 말하고 나중에 멤버 이름을 말하는 것이다.



Date 객체

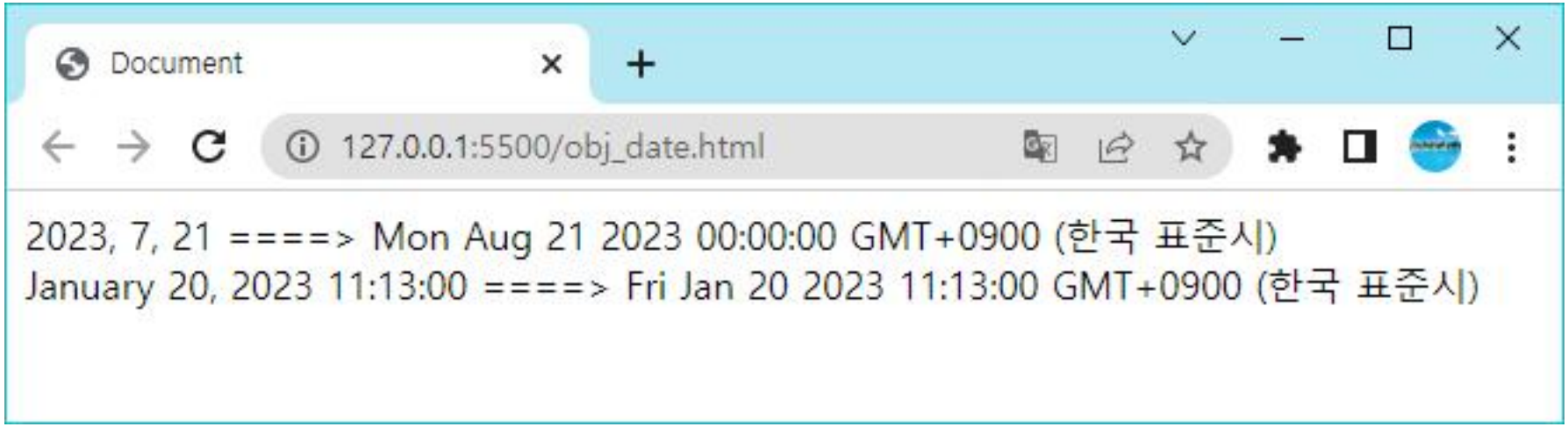
- Date 객체는 날짜와 시간을 저장하고 있는 객체이다. 자바스크립트의 내장 객체 중에서도 Date 객체는 상당히 많이 사용된다. 웹 프로그래밍에서는 날짜를 사용하는 작업이 많기 때문이다.

```
new Date()                                          // 현재 날짜와 시간
new Date(milliseconds)        // 1970/01/01 이후의 밀리초
new Date(dateString)                          // 다양한 문자열
new Date(year, month, date[, hours[, minutes[, seconds[,ms]]]]) // 상세한 날짜 지정
```

```
 let d1 = new Date();                                          // 현재 날짜와 시간
 let d2 = new Date(2023, 7, 21);                               // 2023년 8월 21일, 7월이
아니다.
 let d3 = new Date("August 24, 2023 18:30:00");   // 문자열로도 생성 가능
 let d4 = new Date(2023, 7, 21, 18, 30, 0, 0);    // 시간도 지정 가능
```

예 제

```
...
    <script>
        let d1 = new Date(2023, 7, 21, 0, 0, 0);
        let d2 = new Date("January 20, 2023  11:13:00");
        document.write("2023, 7, 21 ====>    " + d1+"<br>");
        document.write("January 20, 2023  11:13:00 ====>    " + d2+"<br>");
    </script>
```



Date 객체 메소드

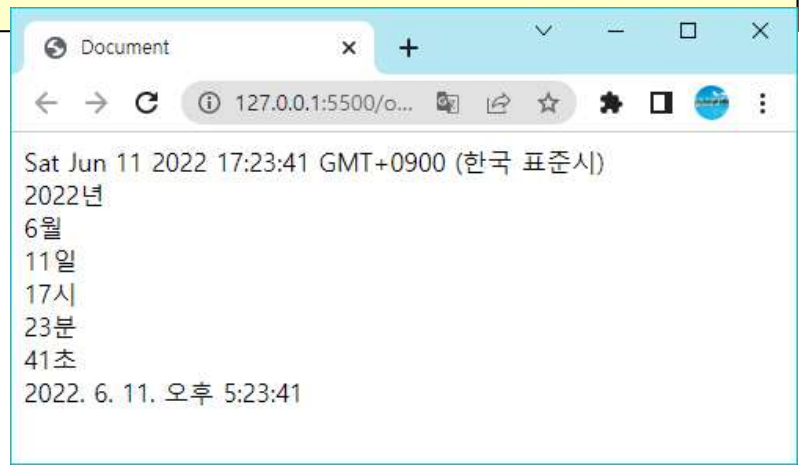
- Date 객체에서 getFullYear(), getMonth(), getDate(), getDay() 메소드를 사용하면 우리는 연도, 월, 일, 시, 분 값을 모두 알 수 있다.

- getDate() (1-31 반환)
- getDay() (0-6 반환)
- getFullYear() (4개의 숫자로 된 연도 반환)
- getHours() (0-23 반환)
- getMilliseconds()(0-999)
- getMinutes()(0-59)
- getMonth()(0-11)
- getSeconds()(0-59)

- setDate()
- setDay()
- setFullYear()
- setHours()
- setMilliseconds()
- setMinutes()
- setMonth()
- setSeconds()

예제

```
<script>
    let today = new Date();
    document.write(today + "<br>");
    document.write(today.getFullYear() + "년<br>");
    document.write(today.getMonth() + 1 + "월<br>");
    document.write(today.getDate() + "일<br>");
    document.write(today.getHours() + "시<br>");
    document.write(today.getMinutes() + "분<br>");
    document.write(today.getSeconds() + "초<br>");
    document.write(today.toLocaleString() + "<br>");
</script>
```

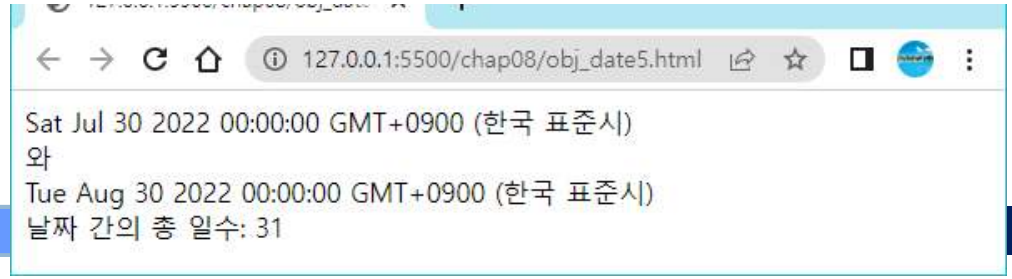


2개의 날짜 비교하기

- Date 객체는 2개의 날짜를 비교하는 데도 사용된다. 두 개의 날짜를 비교하는 것은 생각보다 무척 어렵다. 연도, 월, 일, 윤년 등을 따져야 하기 때문이다.

- ->자바스크립트에서는 유닉스 운영체제에서 사용하였던 방법을 이용한다. 두 개의 날짜를 비교할 때는 먼저 모든 날짜를 1970년 1월 1일 이후의 밀리초로 변환한다. getTime()이라는 메소드를 사용하면 된다.

예 제



- 2022년 7월 30일과 2022년 8월 30일 사이의 총 일수를 계산해보자. 이것을 검사하려면 다음과 같이 한다.

```
<script>
    let date1 = new Date("07/30/2022");
    let date2 = new Date("08/30/2022");
    let diff = date2.getTime() - date1.getTime();
    let diff_days = diff / (1000 * 3600 * 24);

    document.write(
        date1 + "<br> 와 <br>"
        + date2 + " <br> 날짜 간의 총 일수: "
        + diff_days);
</script>
```

참고 사항: 유닉스 시간

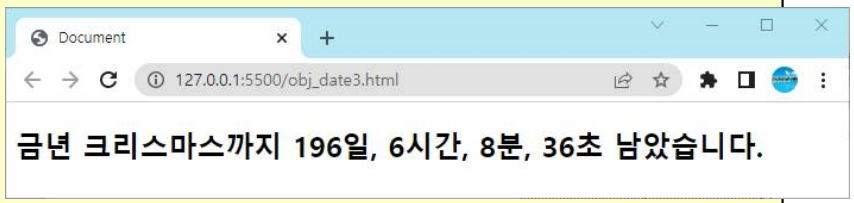
- 유닉스 시간은 1970년 1월 1일을 기준으로 얼마나 많은 밀리초가 흘렀는지를 기준으로 한다. 밀리초는 1초를 1000으로 나눈 숫자여서 아주 많을 거 같지만 컴퓨터는 이 정도 숫자는 순식간에 처리한다.

예제: 카운트다운 타이머 만들기

- 지금부터 특정 날짜까지 남은 날짜를 표시하는 코드를 작성하여 보자. 예를 들어서 크리스마스까지, 남은 날짜를 웹 페이지에 표시하는 예제를 작성하여 보자.

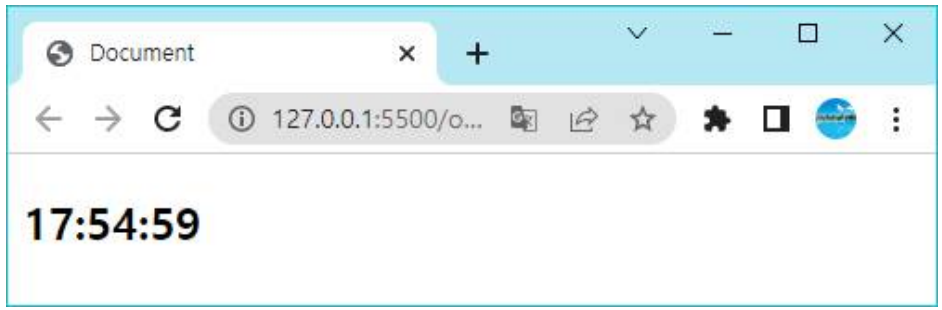
예제

```
<!DOCTYPE html>
<body>
    <h2 id='remaining'></h2>
    <script>
        function datesUntilXmas() {
            let now = new Date();
            let newYear = new Date('December 25, ' + (now.getFullYear()));
            let diff = newYear - now;
            let milliseconds = Math.floor(diff % 1000);
            diff = diff / 1000;
            let seconds = Math.floor(diff % 60);
            diff = diff / 60;
            let minutes = Math.floor(diff % 60);
            diff = diff / 60;
            let hours = Math.floor(diff % 24);
            diff = diff / 24;
            let days = Math.floor(diff);
            let outStr = '금년 크리스마스까지 ' + days + '일, ' + hours + '시간, '
                    + minutes;
            outStr += '분, ' + seconds + '초' + ' 남았습니다.';
            document.getElementById('remaining').innerHTML = outStr;
            // 1초가 지나면 다시 함수를 호출한다.
            setTimeout("datesUntilXmas()", 1000);
        }
        // 타이머를 시작한다.
        datesUntilXmas();
    </script>
</body>
</html>
```



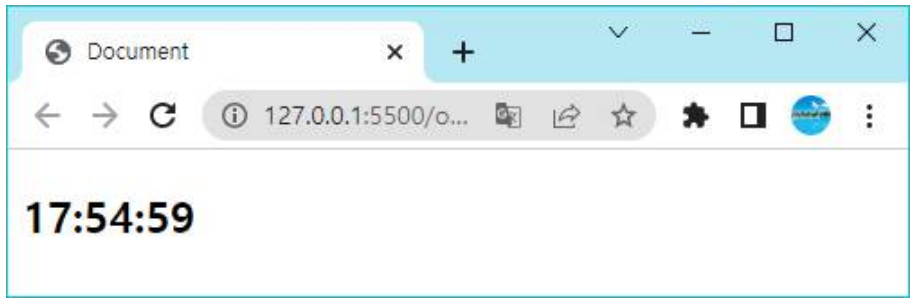
예제: 시계 만들기

- 시계 만드는 것도 타이머와 아주 비슷하다. 새로운 Date 객체를 생성하고 시간, 분, 초를 계산하면 된다.



예 제

```
<h2 id='clock'></h2>
 <script>
     function setClock() {
         let now = new Date();
         let s = now.getHours() + ':' + now.getMinutes() + ':' + now.getSeconds();
         document.getElementById('clock').innerHTML = s;


         setTimeout('setClock()', 1000);
     }
     setClock();
 </script>
```



자바스크립트 디버깅하기

- 이때에는 크롬의 개발자 도구를 이용하여 디버그하면 편리하다. 다음과 같은 디버그 도구들을 사용하도록 하자. 웹페이지에서 마우스 오른쪽 버튼을 누르고 "검사"를 선택하면 개발자 모드가 나타난다.

중간점검

1. Date 객체를 생성하는 몇 가지 방법을 이야기해보자.
2. 첫 번째 날짜에서 두 번째 날짜까지 며칠이나 흘렀는지를 계산하려면 어떻게 하는 것이 좋은가?

String 객체

- String 객체는 텍스트를 저장하고 조작하는 객체로 아주 많이 사용된다. String 클래스의 메소드를 잘 알아두어야 편리하다.
- 웹 프로그래밍에서도 텍스트를 조작해야 하는 경우가 상당히 많기 때문이다

- 자바스크립트에서 문자열(String)에는 2가지 종류가 있다. 문자열 리터럴(String Literal)과 문자열 객체(String Object)가 바로 그것이다.
  - 문자열 리터럴은 따옴표로 생성된다.
  - 문자열 객체는 키워드 new를 이용하여 생성된다.

```
let s1 = "ABCD";
let s2 = new String("ABCD");
```

예제

```
        let sLiteral = "문자열 리터럴";
        sLiteral = sLiteral.concat('123');
        document.writeln(sLiteral  +  '<br>');    //
문자열 리터럴123
```

문자열의 길이와 개별 문자 접근

- String 객체의 속성 length는 문자열의 길이를 나타낸다.

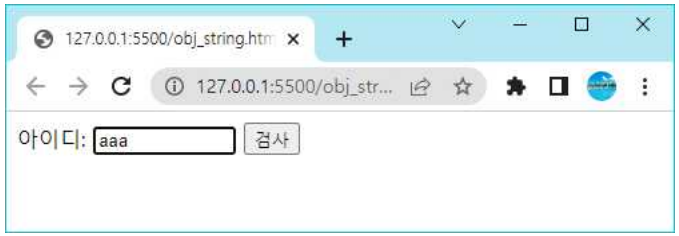
```
let s = "abcdef";
let length = s.length;  // 6이 된다.
```

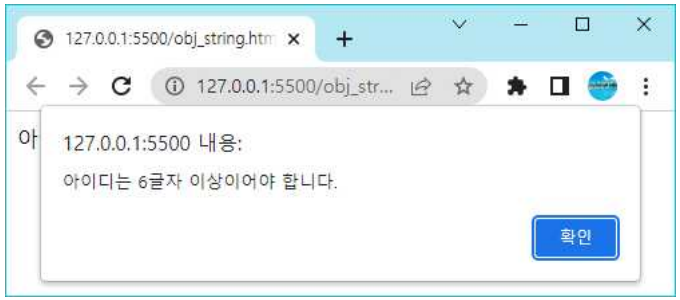
- 문자열 안의 개별적인 문자에 접근하려면 배열처럼 [ ] 안에 인덱스를 넣으면 된다. 인덱스는 0부터 시작한다.

```
let s = "abcdef";
let ch  = s[1];                    // ch는 'b'가 된다.
```

예제

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function checkID() {
            let s = document.getElementById("id").value;
            if (s.length < 6)
                alert("아이디는 6글자 이상이어야 합니다.");
        }
    </script>
</head>
<body>
        아이디: <input type="text" id="id" size=10>
        <button onclick="checkID()">검사</button>
</body>
</html>
```





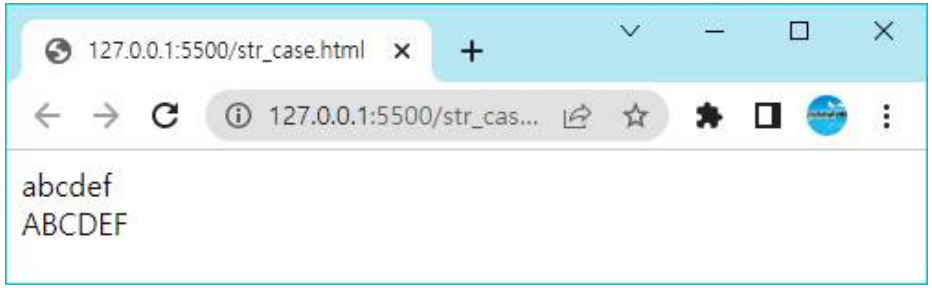
String 객체 메소드

방법	설명
charAt(위치)	지정된 위치의 문자를 반환한다.
indexOf(검색 문자열, 위치)	지정된 숫자 인덱스에서 시작하여 지정된 문자열이 처음 나타나는 인덱스를 반환한다. 찾을 수 없으면 –1을 반환한다.
replace(검색값, 대체값)	지정된 값을 검색하고 대체값으로 바꾸고 새 문자열을 반환한다.
match(RegExp)	지정된 정규식을 기반으로 일치하는 항목을 검색한다.
slice(시작 위치, 종료 위치)	지정된 시작 및 끝 인덱스를 기반으로 문자열의 일부를 추출하고 새 문자열을 반환한다.
split(분리자, 제한 길이)	지정된 분리자에 따라 문자열을 문자열 배열로 분할한다.
substr(시작, 길이)	지정된 시작 위치에서 지정된 문자 수(길이)까지 문자열의 일부를 반환한다.
substring(시작, 끝)	시작 인덱스와 끝 인덱스 사이의 문자열에서 부분 문자열을 반환한다.
toLowerCase()	소문자로 변환한다.
toUpperCase()	대문자로 변환한다.
valueOf()	문자열을 수치값으로 변환한다.

대소문자 변환

- toUpperCase()와 toLowerCase()를 사용하면 문자열의 글자들을 대문자, 혹은 소문자로 변환할 수 있다

```
<script>
    let s = 'aBcDeF';
    let result1 = s.toLowerCase();
    let result2 = s.toUpperCase();
    document.write(result1 + "<br>");
    document.write(result2 + "<br>");
</script>
```



문자열 비교

- 문자열을 비교하는 가장 쉬운 방법은 == 연산자를 사용하는 것이다.

```
if( "apple" == "apple" ){                    // true
        ...
}
```

- 문자열의 사전적인 순서를 알려면 <이나 > 연산자를 사용한다.

```
if( "cat" < "dog" ){                    // true
        ...
}
```

문자열 검색

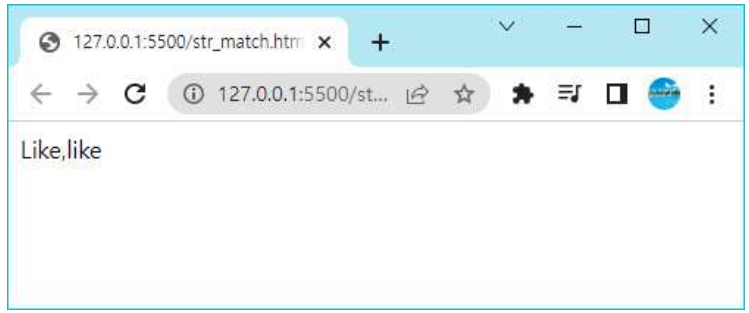
- indexOf() 메소드는 문자열 안에서 주어진 텍스트가 처음 등장하는 위치를 반환한다. 아주 편리한 메소드이다

```
<script>
    let s = "A bad workman always blames his tools.";
    let n = s.indexOf("workman");
    document.write(n + '<br>'); // 6
</script>
```

문자열 매칭

- match() 메소드에서는 정규식(regular expression)을 사용할 수 있다.

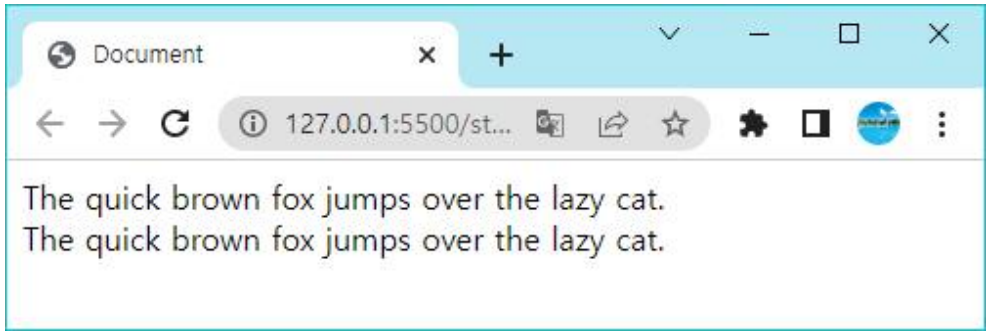
```
<script>

  let str = 'Like father, like son.';
  // like를 찾는다. i와 g는 옵션으로 insensitive, globally를 의미한다.
  result = str.match(/like/ig);
  document.write(result + '<br>');
</script>
```



문자열 대체

- replace() 메소드는 문자열 안에서 주어진 값을 다른 값으로 대체한다. replace()도 정규식을 사용할 수 있어서 강력한 기능 중의 하나이다.

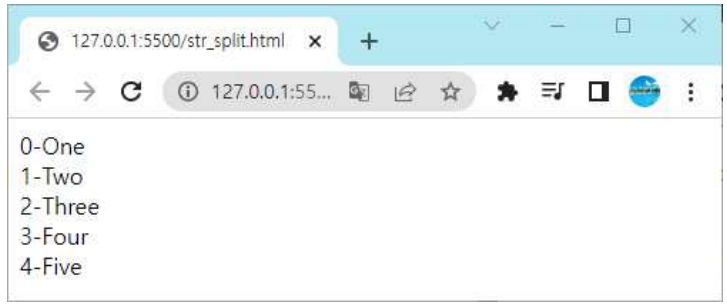
```
<script>
        let p = 'The quick brown fox jumps over the lazy dog.';
        document.write(p.replace('dog', 'cat')+"<br>");
        let regex = /Dog/i;
        document.write(p.replace(regex, 'cat'))+"<br>";
</script>
```



문자열 분할

- split()는 첫 번째 인수를 분리자로 하여서 주어진 문자열을 분리한 후에 각 항목들을 가지고 있는 배열을 반환한다

```
<script>
        let s = "One,Two,Three,Four,Five";
        array = s.split(',');
        for (i = 0; i < array.length; i++) {
            document.write(i + '-' + array[i] + '<br>');
        }
</script>
```



공백 제거

- 문자열 앞뒤에 붙은 공백 문자들을 제거하는데는 trim() 메소드를 사용할 수 있다.

```
let s = "   Hello world!   ";
s = s.trim();                  // s="Hello World!"
```

- trim() 메소드가 왜 필요할까? 사용자가 문자열을 입력할 때 실수로 앞뒤에 공백을 추가하는 일도 매우 흔하게 발생하기 때문이다.