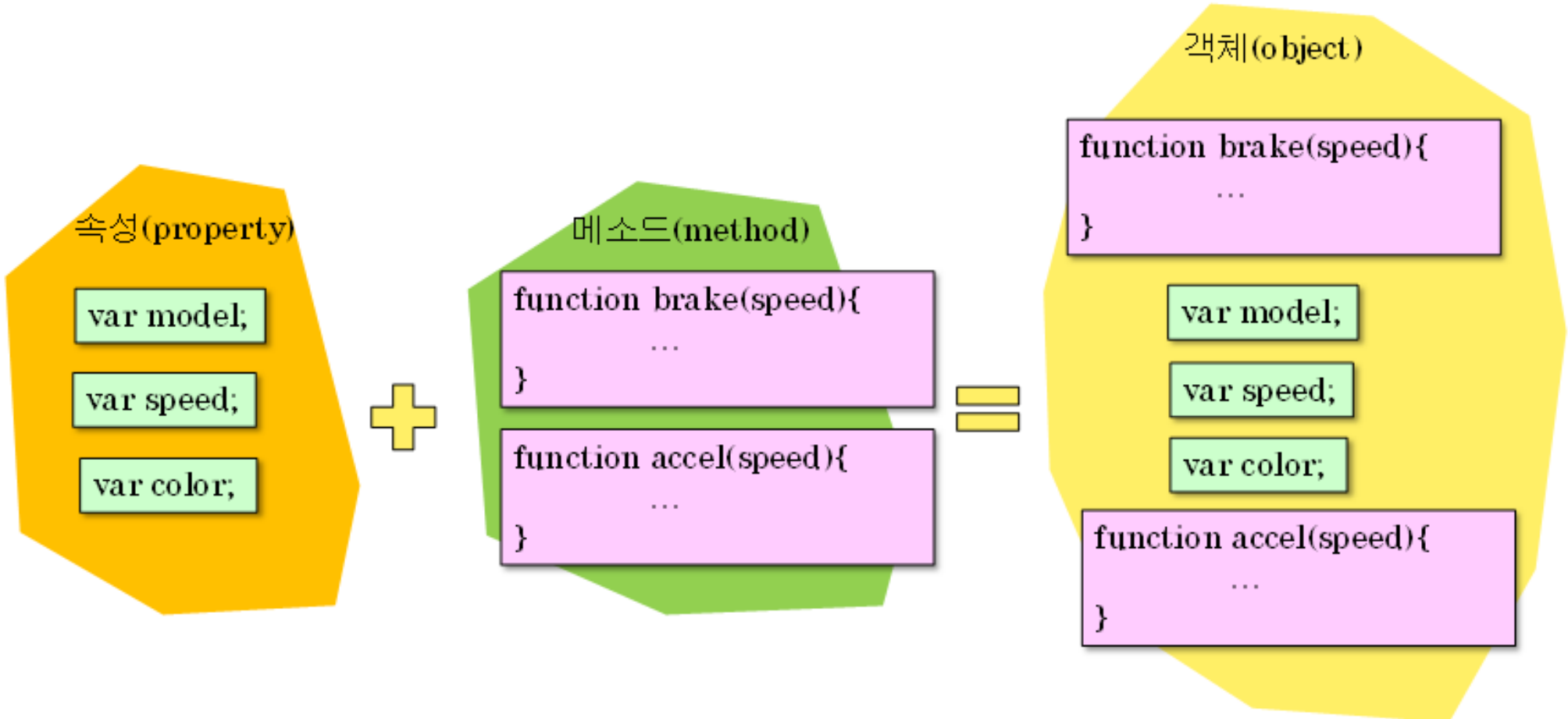
09 자바 스크립트 객체

객체

- **객체(object)**는 사물의 속성과 동작을 묶어서 표현하는 기법
- (예) 자동차는 메이커, 모델, 색상, 마력과 같은 속성도 있고 출발하기, 정지하기 등의 동작도 가지고 있다.



객체의 종류

- 객체의 2가지 종류
  - *내장 객체(bulit-in object):* 생성자가 미리 작성되어 있다.
  - *사용자 정의 객체(custom object):* 사용자가 생성자를 정의한다.

- 내장 객체들은 생성자를 정의하지 않고도 사용이 가능하다. Date, String, Array와 같은 객체들이 내장 객체이다.
- new Array(“apple”, “Orange”);
- new Array();

객체 생성 방법(1/3)

- 객체를 생성하는 2가지 방법
  - 객체를 객체 리터럴로부터 직접 생성한다.
  - 생성자 함수를 이용하여 객체를 정의하고 new 연산자를 통하여 객체의 인스턴스를 생성한다.

- var arr=[2, 4, 6, 10, 34]
- var arr = new Array(1, 3, 4, 5, 6);

객체 생성 방법(2/3)

- 객체 상수로 객체 생성

```
var myCar = {
        model: "520d",
        speed: 60,
        color: "red",
        brake: function () { this.speed -= 10; },
        accel: function () { this.speed += 10; }
};

myCar.color = "yellow";
myCar.brake();
```

객체의 속성

객체의 메소드

생성자를 이용한 객체 생성(3/3)

- 생성자로 객체 생성

```
function Car(model, speed, color) {
    //객체의 속성
    this.model = model;
    this.speed = speed;
    this.color = color;
    //객체의 메소드
    this.brake = function () {
        this.speed -= 10;
    }
    this.accel = function () {
        this.speed += 10;
    }
}
```

생성자도 함수
생성자 이름은 대문자로 시작

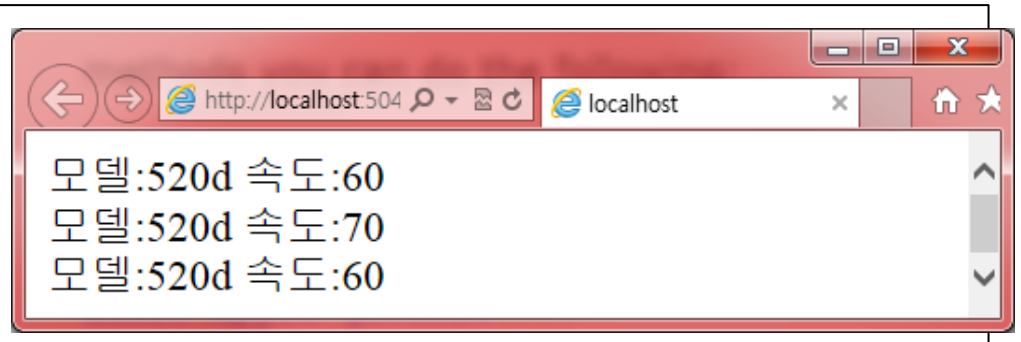
소유자를 의미
this 키워드로 일반 변수와 구별함

생성자 함수를 호출해서
새로운 객체 생성

```
var myCar = new Car(
"소나타", 60, "white");
myCar.color =  “white";
myCar.brake();
```

예제

```
...
<script>
    function Car(model, speed, color) {
        this.model = model;
        this.speed = speed;
        this.color = color;
        this.brake = function () { this.speed -= 10; }
        this.accel = function () { this.speed += 10; }
    }

    myCar = new Car("520d", 60, "red");
    document.write("모델:" + myCar.model + " 속도:" + myCar.speed + "<br>");
    myCar.accel();
    document.write("모델:" + myCar.model + " 속도:" + myCar.speed + "<br>");
    myCar.brake();
    document.write("모델:" + myCar.model + " 속도:" + myCar.speed + "<br>");
</script>
...
```



객체 속성 및 메소드 추가

- 기존에 존재하고 있던 객체에도 속성을 추가할 수 있다.
- 생성자 함수는 변경할 필요가 없다.

```
myCar.turbo = true;
myCar.showModel = function () {
        alert("모델은 " + this.model + "입니다.")
}
```

- 객체 속성 및 메소드 삭제도 가능함

```
delete myCar.turbo;
```

객체 표시 방법

- 속성 표시

```
myCar.model;
myCar["model"];
```

- 배열로 변환 - Object.values()

```
var person = {name:"홍길동", age:30, city:"대전"};
var myArray = Object.values(person);
// myArray는 ["홍길동", 30, "대전"] 이란 배열이 됨
```

- 문자열로 변환 - JSON.stringify()

```
var person = {name:"John", age:50, city:"New York"};
var myString = JSON.stringify(person);
//myString은 {"name":"John","age":50,"city":"New York"} 으로 생성됨
```

자바 스크립트 내장 객체

- Object 객체 – new Object() 대신 {} 사용 가능
- Array 객체 – new Array() 대신 [] 사용 가능
- Date 객체
- Number 객체
- String 객체 - new String() 대신 "" 사용 가능
- Boolean 객체 - new Boolean() 대신 true 나 false 사용 가능
- Function 객체 – new Function() 대신 function(){} 사용 가능
- Math 객체
- …
  - new 키워드를 사용해서 생성 (예: var date = new Date();)
  - Math는 전역 객체 이기 때문에 new 키워드 사용하지 않음

Array 객체

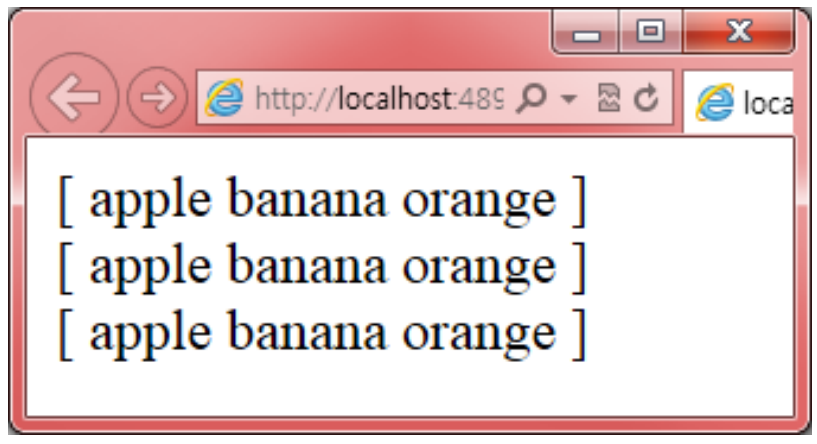
- 배열을 나타내는 객체

```
var myArray = new Array();
myArray[0] = "apple";
myArray[1] = "banana";
myArray[2] = "orange";
```

- 배열의 크기가 자동으로 조절된다. 다른 언어에서는 배열의 크기가 고정되어 있다. 하지만 자바스크립트에서 배열의 크기는 현재 배열의 크기보다 큰 인덱스를 사용하면 자동으로 증가한다.
- 자바스크립트에서는 하나의 배열에 여러 가지 자료형을 혼합해서 저장할 수 있다.
- 즉 하나의 배열에 정수와 문자열을 동시에 저장하는 것이 가능하다.

예제

```
…
<script>
        function printArray(a) {
            document.write("[ ");
            for (var i = 0; i < a.length; i++) {
                document.write(a[i] + " ");
            }
            document.write(" ] <br>");
        }

        var myArray1 = new Array();
        myArray1[0] = "apple";
        myArray1[1] = "banana";
        myArray1[2] = "orange";

        var myArray2 = new Array("apple", "banana", "orange");

        var myArray3 = ["apple", "banana", "orange"];

        printArray(myArray1);
        printArray(myArray2);
        printArray(myArray3);
    </script>
…
```



Array 객체의 메소드

- 속성
  - length
- 메소드

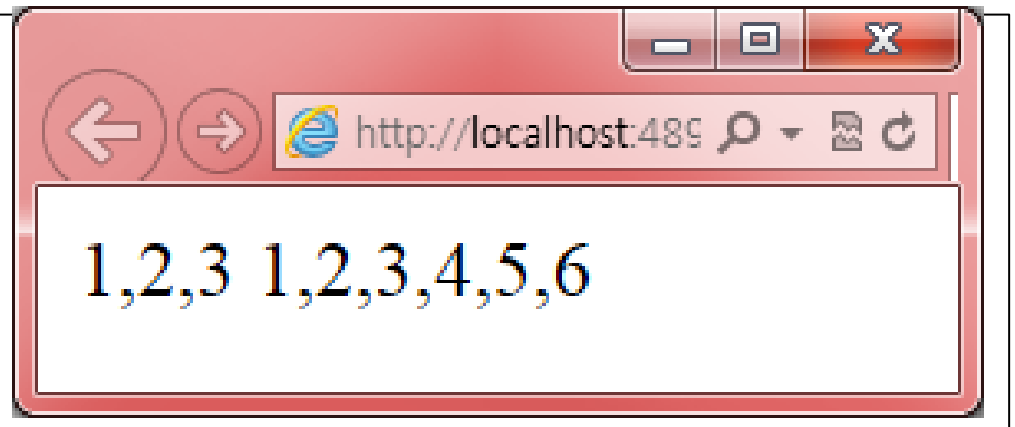
메서드	설명
indexOf(item, start)	배열에서 요소를 찾아 위치를 리턴한다.
lastIndexOf(item, start)	역순으로 요소를 찾아 위치를 리턴한다.
pop()	마지막 요소를 제거하고 리턴한다.
shift()	배열 처음의 원소를 제거하고 리턴한다.
unshift(a,b,c,...)	배열 처음에 요소를 추가한다.
push(a,b,c,...)	배열 끝에 요소를 추가한다.

Array 객체의 메소드

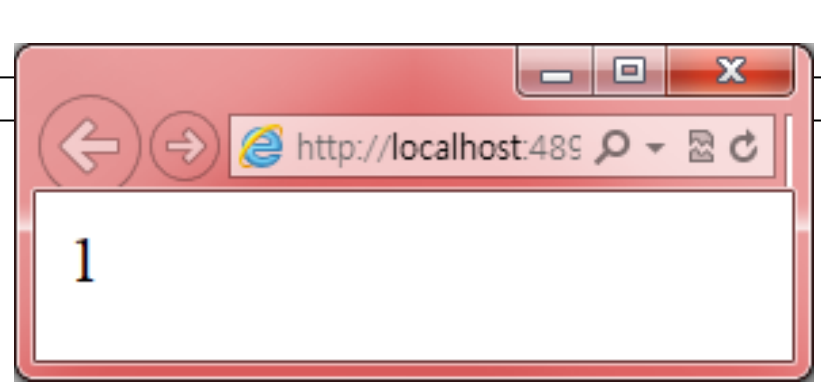
메소드	설명
reverse()	배열을 거꾸로 뒤집는다.
sort(*sortfunction*)	배열을 정렬한다. 인수로 값을 비교하는 함수를 지정할 수 있으며 생략 시 사전 순으로 정렬된다.
slice(startIdx*, endIdx*)	start ~ end 범위의 요소를 따로 떼어내어 새로운 배열을 만든다. (endIdx 생략가능. endIdx는 포함되지 않음)
splice(index, n, a, b, c, ...)	배열 일부를 수정한다. 일정 범위를 삭제하고 새로운 요소를 삽입한다.
a.concat(b,c...)	여러 개의 배열을 합친다.
join(deli)	배열 요소를 하나의 문자열로 합친다. 구분자를 지정할 수 있으며 생략 시 콤마로 구분한다.

예제

```
<script>
    var x = [1, 2, 3];
    var joined = x.concat(y);

    document.writeln(x);      // 출력: 1,2,3
    document.writeln(joined); // 출력: 1,2,3,4,5,6
</script>
```




```
<script>
    var fruits = ["apple", "banana", "grape"];
    document.writeln(fruits.indexOf("banana"));
</script>
```



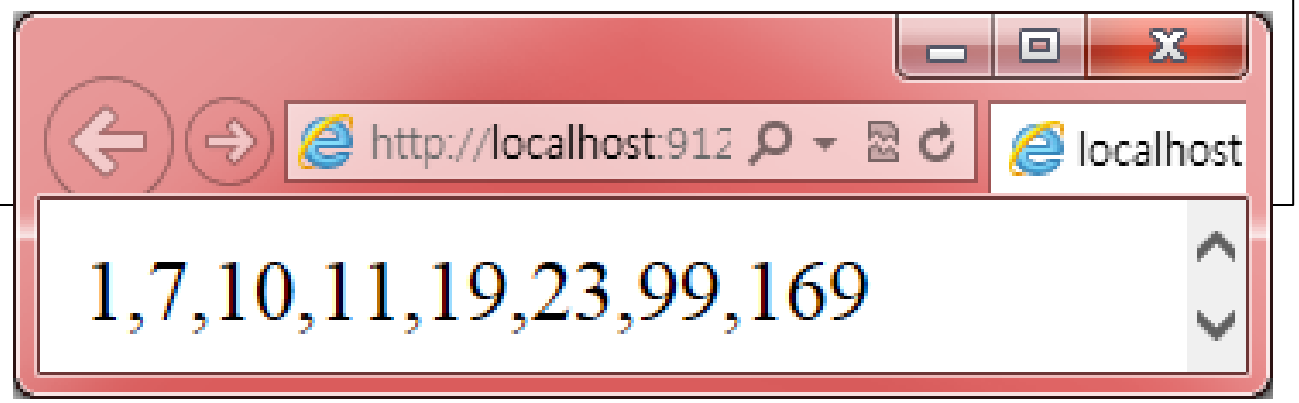
예제

```
<script>
    var numbers = [1, 2, 3, 4, 5];

    numbers.push(6);
    document.writeln(numbers + '<BR>');  // 출력: 1,2,3,4,5,6
    item = numbers.pop();
    document.writeln(numbers + '<BR>');  // 출력: 1,2,3,4,5,
</script>
```

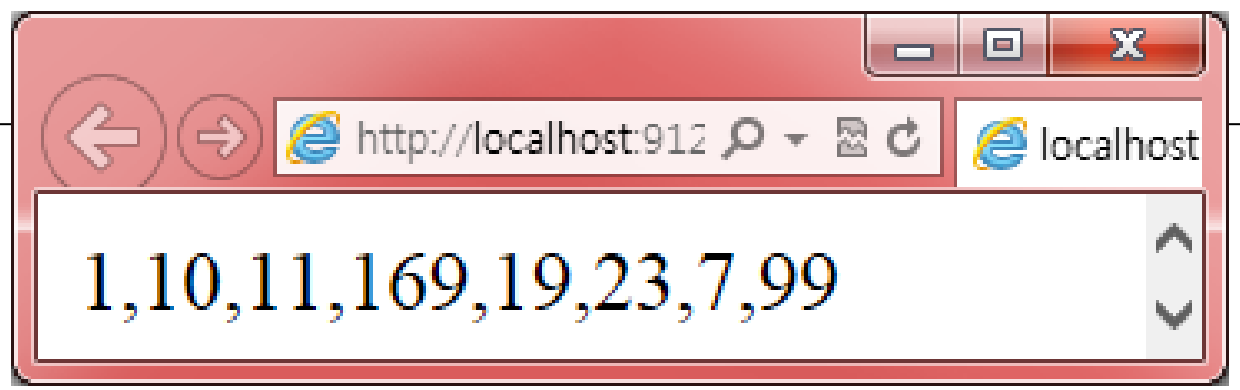
```
<script>
    var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

    var item = numbers.shift();
    document.writeln(item + '<BR>');     // 출력: 1
    document.writeln(numbers + '<BR>');  // 출력: 2,3,4,5,6,7,8,9,10
</script>
```

예제

```
<script>
    var myArray = [10, 7, 23, 99, 169, 19, 11, 1];
    myArray.sort()
    document.writeln(myArray);
</script>
```




```
<script>
    var myArray = [10, 7, 23, 99, 169, 19, 11, 1];
    myArray.sort(function (a, b) { return a - b });
    document.writeln(myArray);
</script>
```



Array 객체 문제

1. 사람 이름을 계속 입력 받아 배열에 저장하고 그 저장된 이름을 출력하는 프로그램을 작성하시오.(단, 입력은 prompt 명령을 이용하고, 입력의 마지막은 공백문자를 입력하거나 "취소" 버튼을 눌렀을 때로 한다. "취소" 버튼은 null 이 입력될 때이다.)

2. 서로 중복되지 않은 정수 5개를 입력 받아 출력하는 프로그램을 작성하시오.

Date 객체

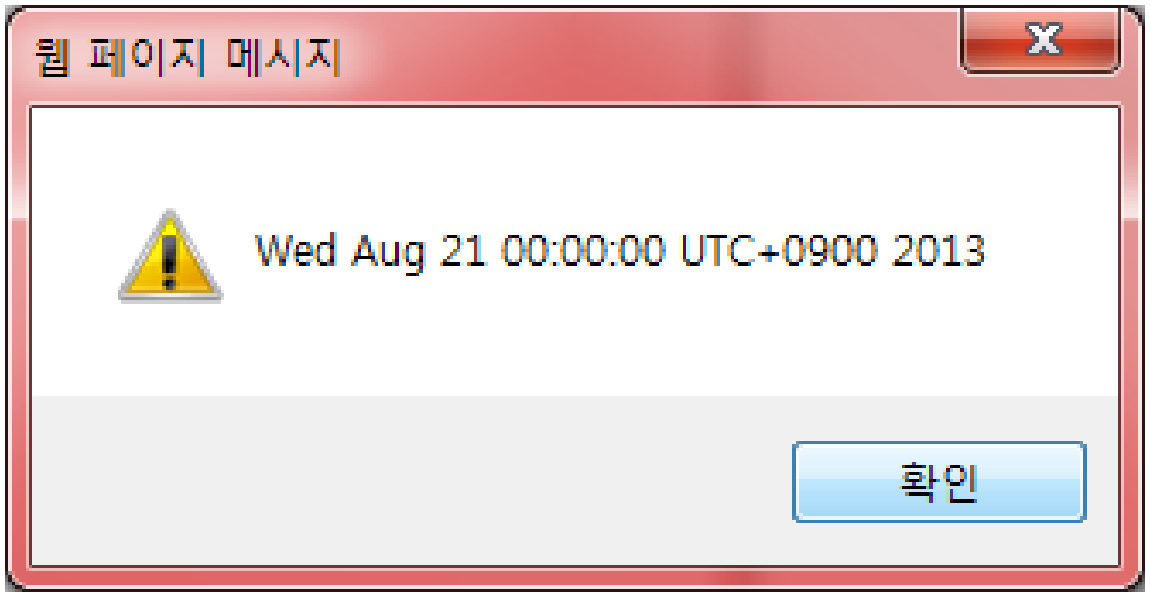
- Date 객체는 날짜와 시간 작업을 하는데 사용되는 가장 기본적인 객체
  - new Date() // 현재 날짜와 시간
  - new Date(milliseconds) //1970/01/01 이후의 밀리초
  - new Date(dateString)
  - new Date(year, month, date[, hours[, minutes[, seconds[,ms]]]])

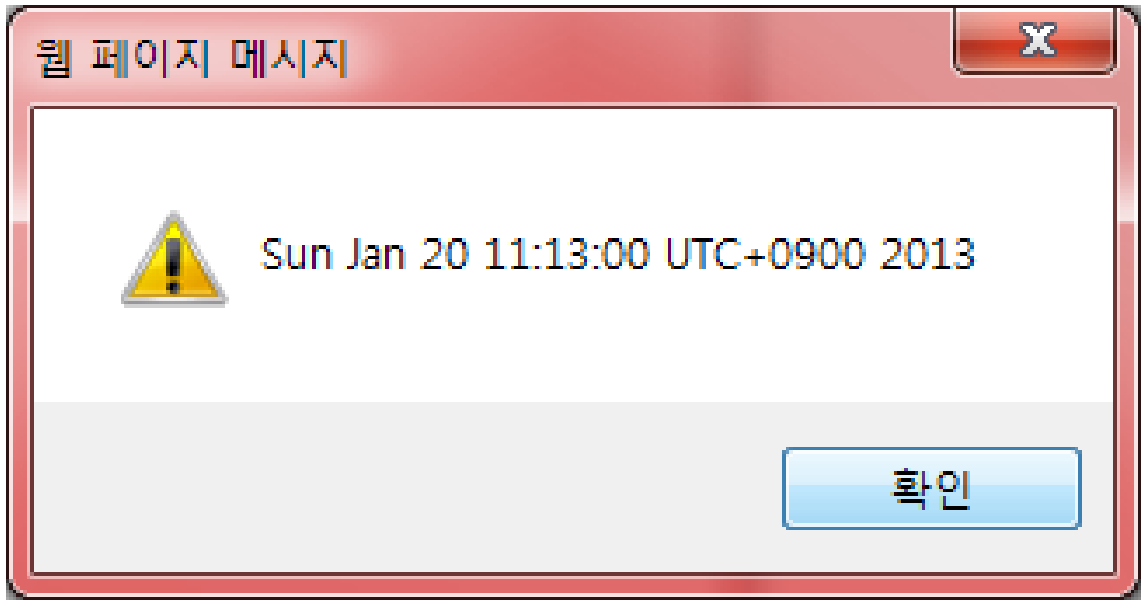
- Month는 0부터 시작함
- Year는 두자리로도 표시 가능

- UTC
- 1972년 1월 1일부터 시행된 국제 표준시이다..
- UTC는 그리니치 평균시(GMT)로 불리기도 하는데, UTC와 GMT는 초의 소숫점 단위에서만 차이가 나기 때문에 일상에서는 혼용되어 사용된다. 기술적인 표기에서는 UTC가 사용된다.

예제

```
<script>
    var d1 = new Date(2013, 7, 21, 0, 0, 0);
    var d2 = new Date("January 20, 2013  11:13:00");
    alert(d1);
    alert(d2);
</script>
```





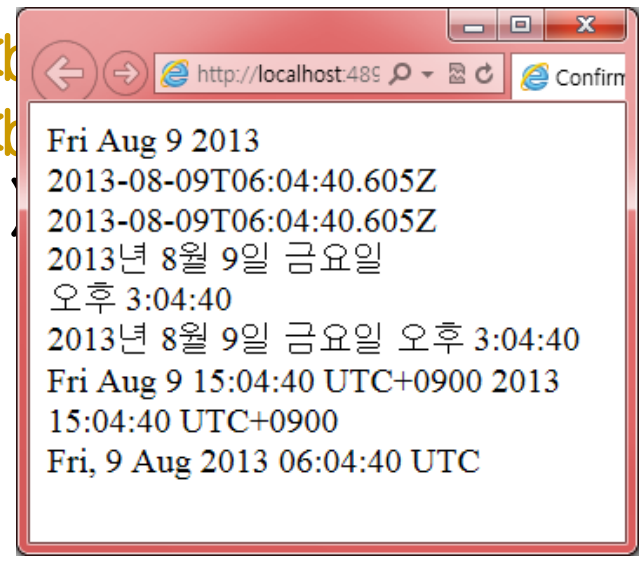
Date 객체의 메소드

get 함수명	반환값	set 함수명
getDay()	0(일요일) ~ 6(토요일)	setDay(day)
getDate()	1 ~ 31	setDate(date)
getMonth()	0 ~ 11	setMonth(month-1)
getFullYear()	4개의 숫자로 된 연도	setYear(year)
getHours()	0 ~ 23	setHours(hours)
getMinutes()	0 ~ 59	setMinutes(minutes)
getSeconds()	0 ~ 59	setSeconds(seconds)
getMilliseconds()	0 ~ 999	setMilliseconds(millisec)
getTime()	경과시간(milliseconds 단위)	setTime(millisec)

getTime의 값 milisec에서 1000으로 나누면 실제 초를 얻을 수 있다

예제

```
<script>
    var today = new Date();
    document.write(today.toDateString() + "<br>");
    document.write(today.toISOString() + "<br>");
    document.write(today.toJSON() + "<br>");
    document.write(today.toLocaleDateString() + "<b
    document.write(today.toLocaleTimeString() + "<b
    document.write(today.toLocaleString() + "<br>"
    document.write(today.toString() + "<br>");
    document.write(today.toTimeString() + "<br>");
    document.write(today.toUTCString() + "<br>");
</script>
```



toISOString()

ISO-8601 and the format is: YYYY-MM-DDTHH:mm:ss.sssZ
시간대는 항상 UTC이며 출력에서 접미사 Z로 표시됩니다

toJSON()

SO-8601 standard: YYYY-MM-DDTHH:mm:ss.sssZ

Date객체

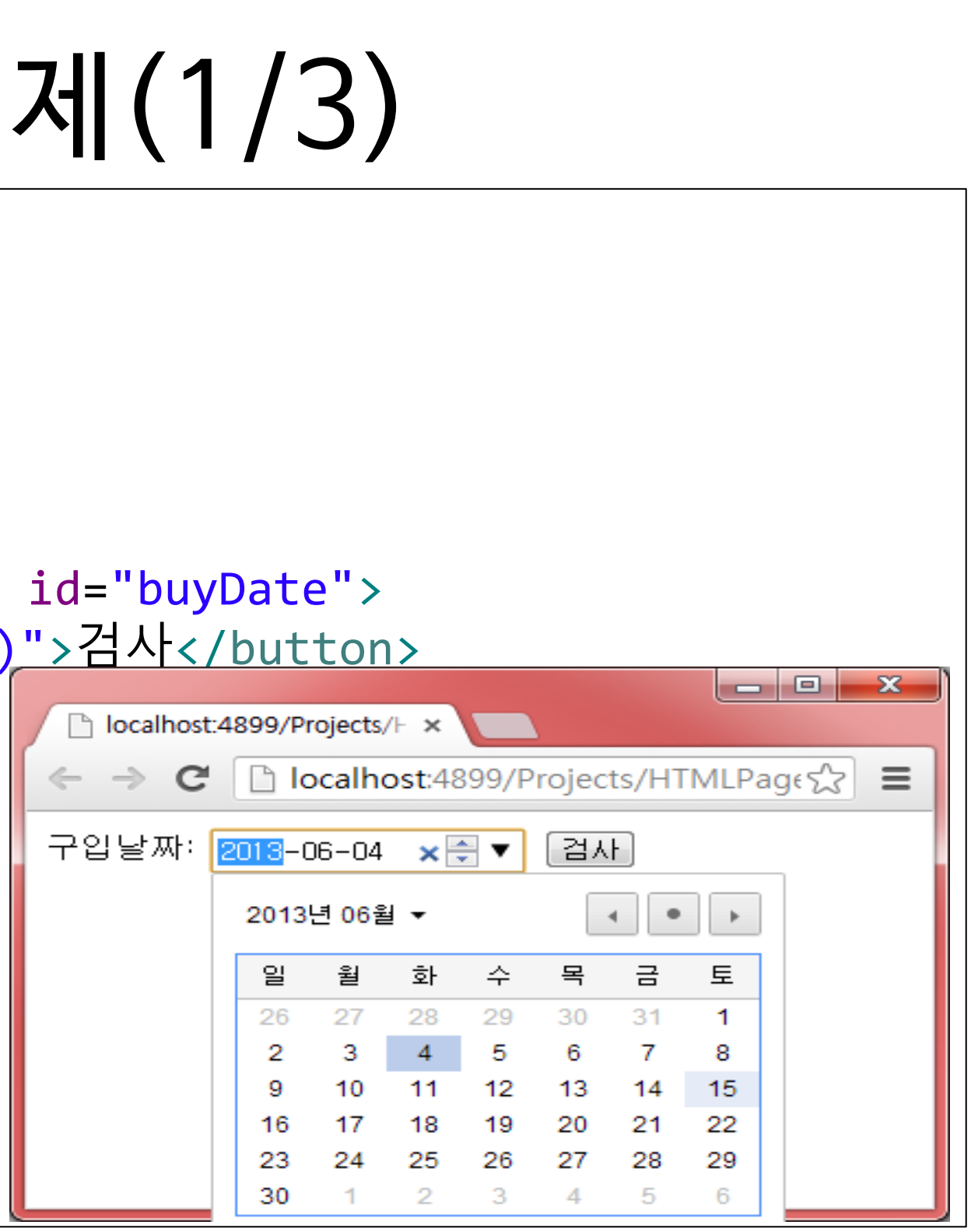
- 날짜수 계산 시 - getTime() 이용
  - (new Date()).getTime() : 오늘 날짜를 milisec으로 반환
  - milisec에서 일자로 변환하기 (/ 1000 / 60 / 60/ 24)

- ex) 오늘의 30일 후 계산하기

```
//오늘날짜를 milisec으로 계산
var currTime = new Date().getTime();

//30일을 milisec값으로 계산
var addTime = 30 * 24 * 60 * 60 * 1000;

//30일 후의 날짜
var resultDate = new Date(currTime + addTime);

alert("오늘로 부터 30일 후의 날짜: " + resultDate);
```

Date 예제(1/3)

```
<script>
    function checkDate() {
        … time)/1000/60/60/24);
    }
</script>
<body>
    구입날짜: <input type="date" id="buyDate">
    <button onclick="checkDate()">검사</button>
</body>
```



Date 예제(2/3)

1. id가 pdate인 엘리먼트의 값을 가져와 변수에 대입한다
2. 1번의 값으로 Date객체를 생성한다 - -pday
3. 오늘의 값으로 Date객체를 생성한다 -today
4. getTime()을 이용하여 3번에서 2번을 뺀 값을 계산 – milisec값
5. 4번의 값으로 하루의 값(milisec)을 계산하여 나눈다
   time/(1000*60*60*24) // time에서 일을 구할때는 나누기
6. 5번의 값이 7보다 큰지 안큰지 비교
   7보다 크면 교환기간이 지났습니다
   아니면 교환 가능합니다

Date 예제(3/3)

```
<script type="text/javascript">
    function checkDate() {
        var pdate = document.getElementById("pdate").value;
        var pTime = new Date(pdate).getTime();//구입날짜 milisec단위

        var currTime = new Date().getTime();//오늘날짜 milisec단위

        //일자계산하기
        var result = (currTime - pTime)/1000/60/60/24;
        if(result > 8 ) {
            alert("교환기간이 지났습니다. 교환기간은 7일 입니다.");
        } else {
            alert("교환가능합니다.");
        }
    }
</script>
<body>
    구입날짜 : <input type="date" id="pdate">
    <input type="button" value="검사" onclick="checkDate();">
</body>
```

String 객체

- 속성
  - length : 문자열의 길이.
  - var str = "I Love Korea "; str.length; ➔ 13

메서드	설명
charAt(index)	index 위치의 문자를 구한다. index가 문자열의 범위를 벗어나면 빈 문자열이 리턴된다.
charCodeAt(index)	index 위치의 문자에 대한 유니코드를 구한다.
indexOf(searchvalue, start)	부분 문자열의 위치를 검색한다. start는 검색 시작 위치이며 생략시 0이 적용되어 처음부터 검색한다. 없을 경우 -1을 리턴한다.
lastIndexOf(searchvalue, start)	부분 문자열의 위치를 역방향에서 검색한다. start는 검색 시작 위치이며 생략 시 문자열의 제일 끝이 적용된다. 없을 경우 -1을 리턴한다.
concat(s1, s2, ...)	여러 개의 문자열을 연결한다. + 연산자와 동일하다.
trim()	앞 뒤의 공백을 제거한다.

String 메소드

메소드	설명
toLowerCase()	소문자로 변환한다
toUpperCase()	대문자로 변환한다
replace(search, value) replaceAll(search, value)	문자열을 대체한다. 정규식도 사용 가능
search(searchvalue)	부분 문자열 또는 정규식을 검색하여 그 위치를 리턴한다.
match(regexp)	정규식으로 검색하여 일치하는 결과를 배열로 리턴한다. 발견되지 않으면 null을 리턴한다.
slice(start, end)	start 위치에서 end 위치까지 부분 문자열을 추출한다. 음수로 끝에서부터 위치를 지정할 수 있다.
substring(from-index, to-index)	두 위치 사이의 부분 문자열을 추출한다. to를 생략하면 뒤쪽 모든 문자열을 추출한다. (to-index는 포함하지 않음)
substr(start, length)	start에서 시작하여 length 길이만큼 부분 문자열을 추출한다. 길이를 생략하면 뒤쪽 모든 문자열을 추출한다.
split(separator, limit)	구분자로 구분된 문자열을 분리하여 배열로 리턴한다. limit는 최대 몇 개까지 리턴할 것인가를 지정한다.

글자 위치 찾기 및 추출

앞에서 찾기

뒤에서 찾기

0	1	2	3	4	5	6	7	8	9	10	11	12	13
우	리	나	라		대	한	민	국		좋	은	나	라

찾음 찾음

```
var s = "우리나라 대한민국 좋은나라";
s.charAt(2); //출력값 : “나”
s.indexOf("나라"); //출력값 : 2
s.lastIndexOf("나라"); //출력값 : 12
s.substring(3, 8); //출력값 : “라 대한민”
s.substring(3); //출력값 : “라 대한민국 좋은나라”
s.substr(3, 4); //출력값 : “라 대한”
s.slice(3, 6); // 출력값 : “라 대”
s.slice(3, -2); // 출력값 : “라 대한민국 좋은”
```

유니코드란

- 유니코드란 전 세계적으로 사용하는 모든 문자 집합을 하나로 모은 것이다. 유니코드 1.0.0은 1991년 8월 제정되었으며, 그 후 약 5년이 지나서야 유니코드 2.0.0에 한글 11,172자가 모두 포함되었다. 현재 버전은 2010년 10월 11일 제정된 6.0이다.
- 유니코드 값을 나타내기 위해서는 코드 포인트(code point)를 사용하는데, 보통 U+를 붙여 표시한다. 예를 들어, 'A'의 유니코드 값은 U+0041로 표현한다(₩u0041로 표기하기도 함). 유니코드는 공식적으로 31비트 문자 집합이지만 현재까지는 21비트 이내로 모두 표현이 가능하다. 유니코드는 논리적으로 평면(plane)이라는 개념을 이용하여 구획을 나누며, 평면 개수는 0번 평면인 기본 다국어 평면(BMP; Basic Multilingual Plane)에서 16번 평면까지 모두 17개이다. 대부분의 문자는 U+0000~U+FFFF 범위에 있는 기본 다국어 평면에 속하며, 일부 한자는 보조 다국어 평면(SMP, Supplementary Multilingual Plane)인 U+10000~U+1FFFF 범위에 속한다. 이 중 한글은 U+1100~U+11FF 사이에 한글 자모 영역, U+AC00~U+D7AF 사이의 한글 소리마디 영역에 포함된다.
- **유니코드의 인코딩 방식**
- 유니코드의 인코딩 방식으로는 코드 포인트를 코드화한 UCS-2와 UCS-4, 변환 인코딩 형식(UTF, UCS Transformation Format)인 UTF-7, UTF-8, UTF-16, UTF-32 인코딩 등이 있다. 이 중 ASCII와 호환이 가능하면서 유니코드를 표현할 수 있는 UTF-8 인코딩이 가장 많이 사용된다. UTF-8은 코드 포인트 범위에 따라 다음 표에서 보는 바와 같이 인코딩 방식이 다르다. 다음 표는 코드 포인트 범위에 따른 UTF-8 인코딩 방식을 보여준다.

예제

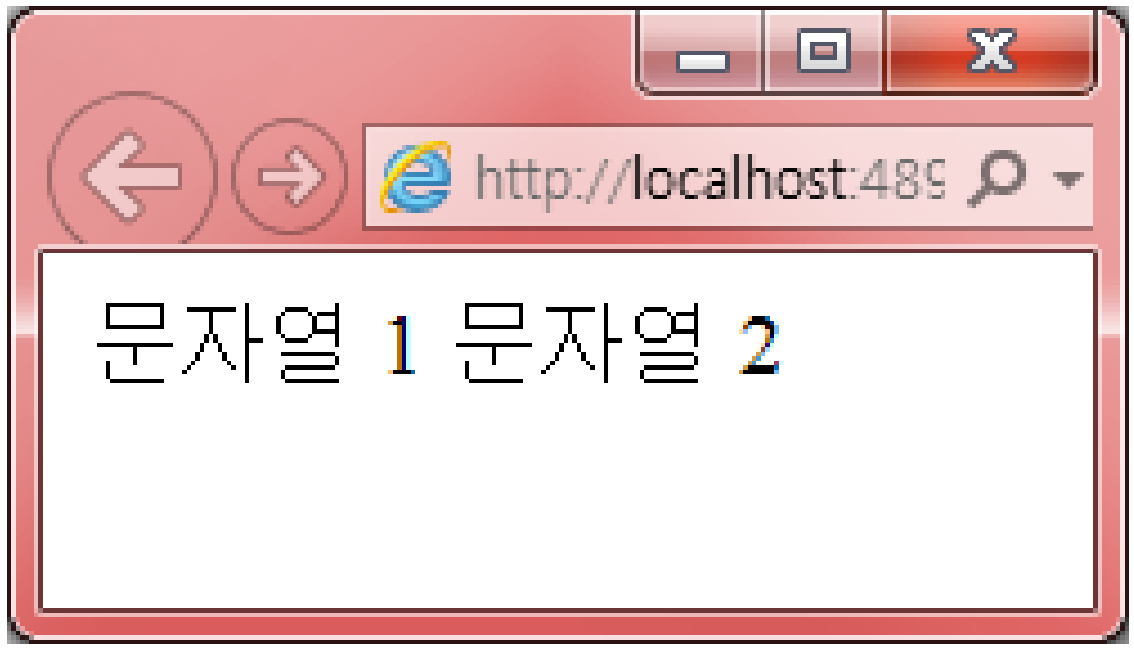
```
<script>
var s = '독도는 일본땅';
document.write(s + "이라는 말도 안되는 소리를 바꿔줍니다.<br>");
document.write("검색 위치 : " + s.search("일본") + "<br>");
document.write("진실된 말 : " + s.replace("일본", "한국") + "<br>");
</script>
```

예제

```
<script>
    var s = 'aBcDeF';
    var result1 = s.toLowerCase();
    var result2 = s.toUpperCase();
    document.write(result1); // 출력: abcdef
    document.write(result2); // 출력: ABCDEF
</script>
```

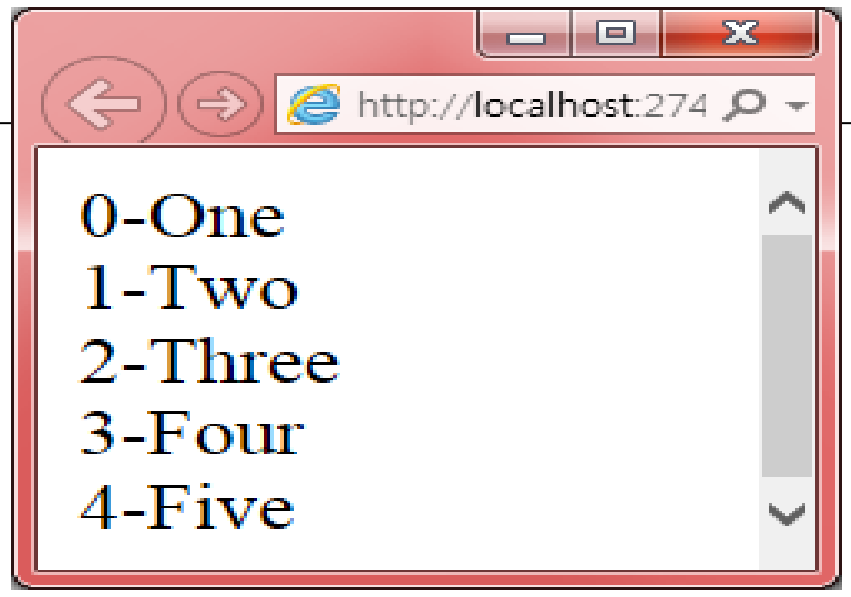
예제

```
<script>
    var s1 = " 문자열 1 ";
    var s2 = " 문자열 2 ";

    s3 = s1.concat(s2);
    document.writeln(s3 + '<br>'); // "문자열 1  문자열 2"
</script>
```



예제

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



String 객체 문제

- 주민등록번호를 입력 받아 생년월일과 성별을 출력하는 프로그램을 작성하시오.(입력은 prompt로 입력받는다.)
  - 예) 주민등록번호를 110326-4432618로 입력 받은 경우
  - 생일 : 2011년 3월 26일
  - 성별 : 여자
  - 나이 : 10살

- 주민등록번호를 입력 받아 주민등록번호의 유효성을 검사하는 프로그램을 작성하시오.(ABCDEF-GHIJKLM)
  1. A*2 + B*3 + … + H*9 + I*2 + … + L*5 의 총합을 구한다.
  2. 1번의 합을 11로 나눈 나머지를 구한다.
  3. 11에서 2번의 결과를 뺀다.
  4. 3번의 결과가 0~9이면 값 그대로, 10이면 0, 11이면 1로 변환
  5. 4번의 결과와 M자리의 값이 같으면 맞는 번호이다.

Math 객체 속성

속성	설명
E	오일러의 상수(약 2.718)
LN2	자연 로그(밑수:2)(약 0.693)
LN10	자연 로그(밑수:10)(약 2.302)
PI	파이 상수(약 3.14)
SQRT1_2	½의 제곱근(약 0.707)
SQRT2	2의 제곱근(약 1.414)

Math 객체 메소드

메소드	설명
abs(x)	절대값
acos(x), asin(x), atan(x)	아크 삼각함수
ceil(x), floor(x)	실수를 정수로 올림, 내림 함수
cos(x), sin(x), tan(x)	삼각함수
exp(x)	지수함수
log(x)	로그함수
max(x,y,z,...,n)	최대값
min(x,y,z,...,n)	최소값
pow(x,y)	지수함수 X^y
random()	0과 1사이의 난수값 반환
round(x)	반올림
sqrt(x)	제곱근

원하는 범위의 랜덤값 만들기

- 0<= Math.random() <1
- Math.floor(Math.random() * (최대값 – 최소값+1) +최소값);
- Math.round(Math.random() * (최대값 – 최소값) + 최소값);

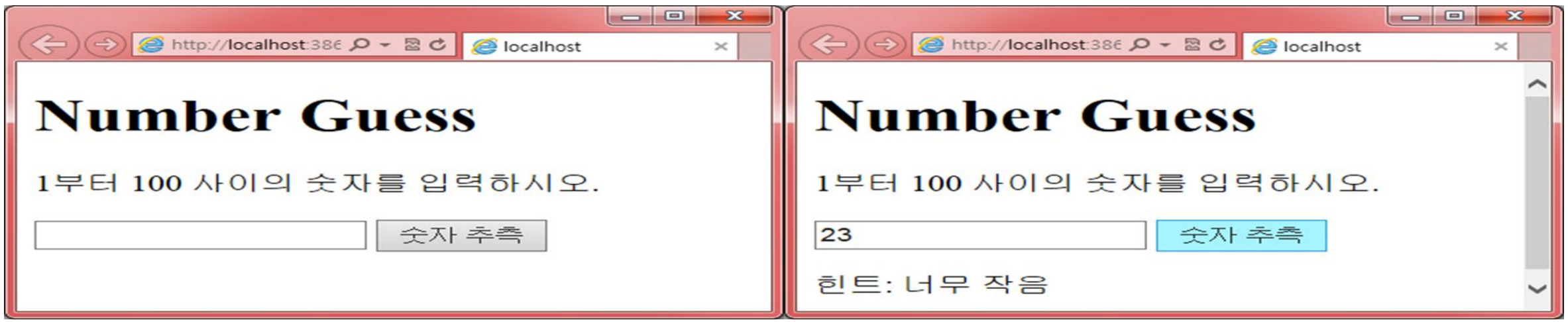
  - 예) 1부터 10까지의 랜덤수 만들기

```
var ranNum = Math.floor(Math.random() * (10 - 1 + 1) + 1);
document.write(ranNum);
```

- Math.random() *10 -> 0~ 9
- Math.random() *10+1 - > 1~ 10
- Math.random() *20 + 11 -> 11 ~30

예제

```
<h1>Number Guess</h1>
    <p>1부터 100 사이의 숫자를 입력하시오.</p>
    <input id="number" type="text">
    <button type="button" onclick="test()">숫자 추측</button>
    <p id="message"></p>
```



Math 객체 문제

1. 1 ~ 10 사이의 난수를 발생 후 사용자가 이 값을 맞추는 프로그램을 작성하시오.

2. 가위 바위 보 게임을 할 수 있는 프로그램을 작성하시오. (단, 컴퓨터는 랜덤을 이용하고, 사용자는 prompt로 입력 받아서 처리)

3. 로또 번호를 생성하는 프로그램을 작성하시오.(1번 ~ 45번 중 6개의 번호를 추첨)