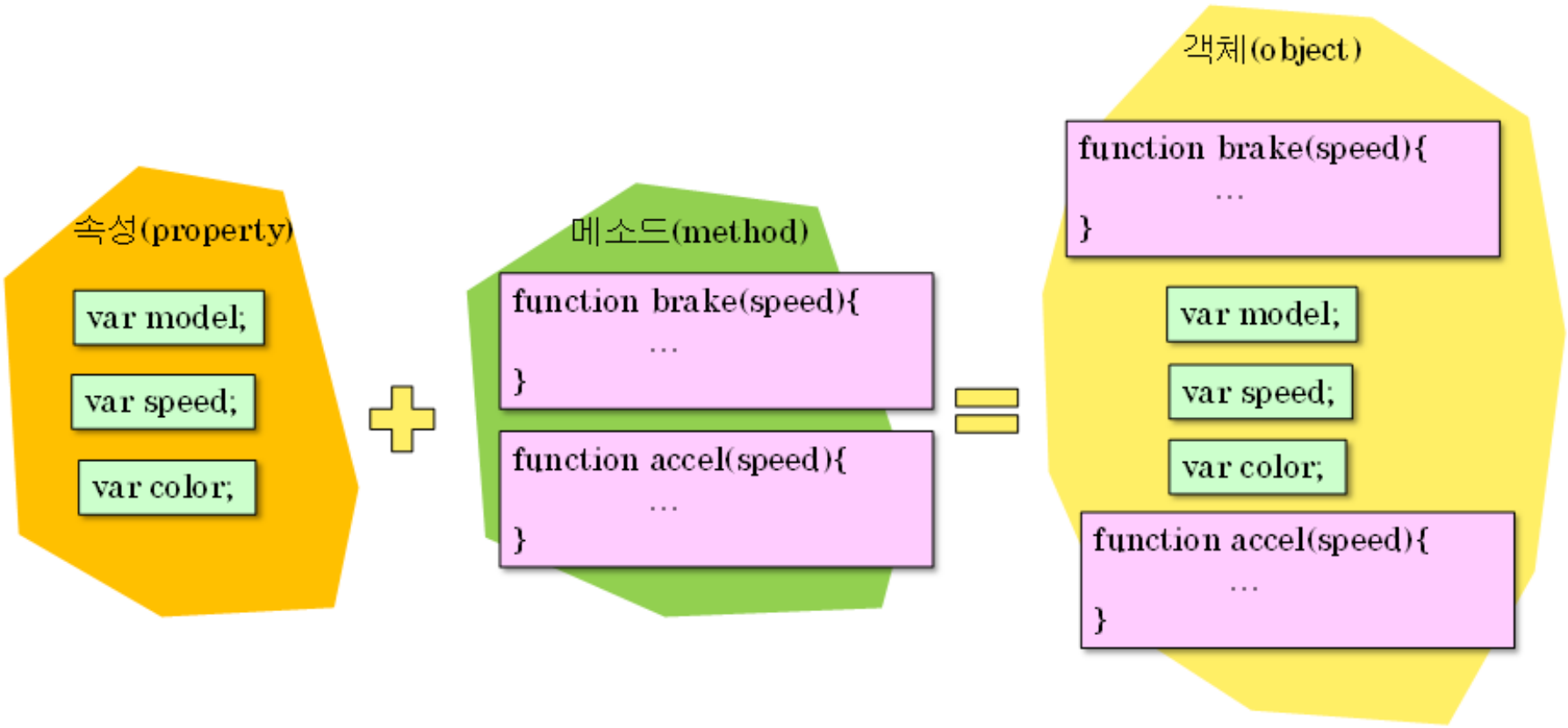
CHAPTER 9.
자바 스크립트 객체

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.



객체

- 객체는 여러 종류의 데이터를 처리하는 일종의 복합 데이터 타입이다.
  - 단일 종류의 정보만을 나타내는 숫자, 스트링, boolean 타입은 원시 데이터 타입이라고도 한다.

- 객체를 사용하면
  - 논리적으로 연관된 변수들을 하나의 그룹으로 묶을 수 있으며
  - 다른 코드가 접근해서는 안 될 변수에 접근하는 것을 막을 수 있다.

- 객체는 관련성이 높은 데이터(property)와 데이터 조작을 위한 동작(method)을 하나로 묶은 데이터 구조이다 → 즉, 객체는 단일 구조로 합쳐진 여러 개의 변수와 여러 개의 함수이다.

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.

객체 생성 방법

- 객체를 생성하는 2가지 방법
  - 객체를 객체 상수로부터 직접 생성한다.
  - 생성자 함수를 이용하여 객체를 정의하고 new를 통하여 객체의 인스턴스를 생성한다.

객체 상수로부터 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

myCar라는 이름의 객체가 하나 생성될 뿐이다 (일반적인 변수 선언 방식과 유사)

생성자를 이용한 객체 생성

생성자도 함수이다.

생성자 이름은 항상 대문자로 한다.

```
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;
    }
}
```

this 키워드로 일반 변수와 객체 속성을 구별한다.

객체의 속성

객체의 메소드

- Property: model, speed, color
- Method: brake(), accel()

교재 320페이지의 함수선언 참조

생성자를 이용한 객체 생성

new 연산자는 새로운 객체를 만든다.

이 인수들로 객체가 초기화된다.

이 변수에 객체가 저장된다.

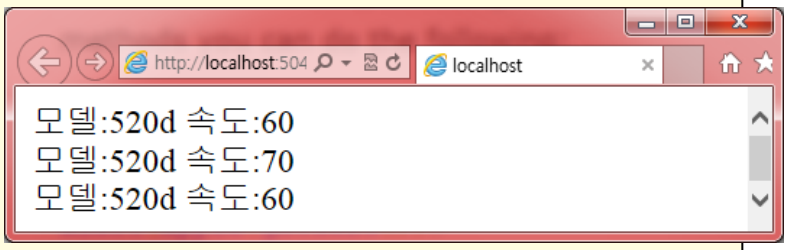
```
var myCar = new Car("520d", 60, "white");
myCar.color = "yellow";
myCar.brake();
```

생성자를 이용한 객체 생성

- 생성자의 가장 큰 임무는 객체의 속성들을 초기값으로 설정하는 것
- 모든 사용자 정의 객체는 생성자가 필요하다!
  - 생성자가 없으면 일반적으로 객체에 property를 부여할 수가 없다 → property가 없으면 객체가 쓸모가 없다.
- "this" 키워드로 일반적인 변수와 객체 property를 구분한다.
  - "this": 객체 참조를 위해 사용되는 자바스크립트 키워드
  - 동일한 객체로부터 해당 객체를 참조하는 과정에서 사용
  - "this"가 객체의 소유자를 의미한다고 상상해볼 것 → "this.model"은 현재 관심 대상인 객체의 소유자가 관심의 초점인 객체의 "model"이라는 property에 접근방법이라고 이해할 수 있음
  - 각 인스턴스는 property에 대하여 **자신만의 복사본**을 갖는데, "this" 키워드를 이용하여 특정 인스턴스만의 복사본임을 명시적으로 표시

객체 생성 예제

```
<!DOCTYPE html>
<html>
<body>
   <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>
</body>
</html>
```



객체에 속성과 메소드 추가

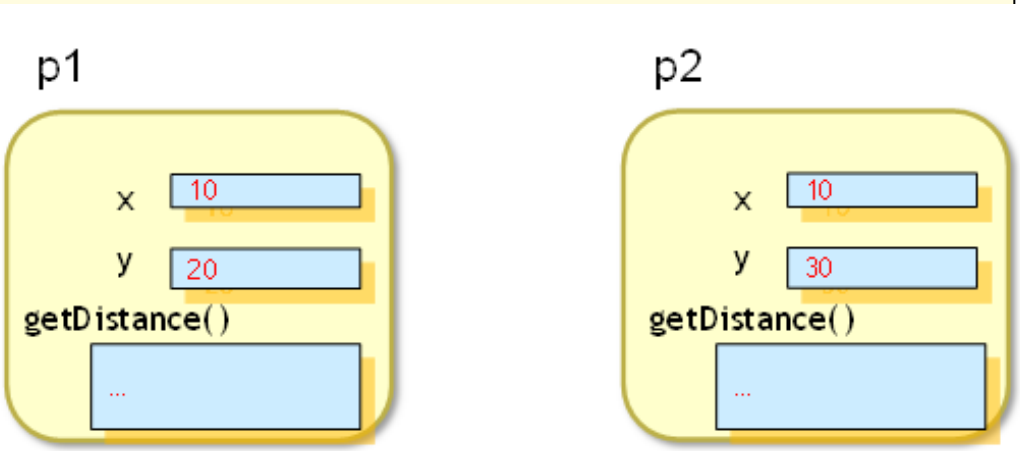
- 기존에 존재하고 있던 객체에도 속성을 추가할 수 있다.
- 생성자 함수는 변경할 필요가 없다.

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

프로토타입

- 자바 스크립트에서 메소드를 여러 객체가 공유하려면 어떻게 해야 하는가?
- 현재는 메소드를 공유할 수 없다.

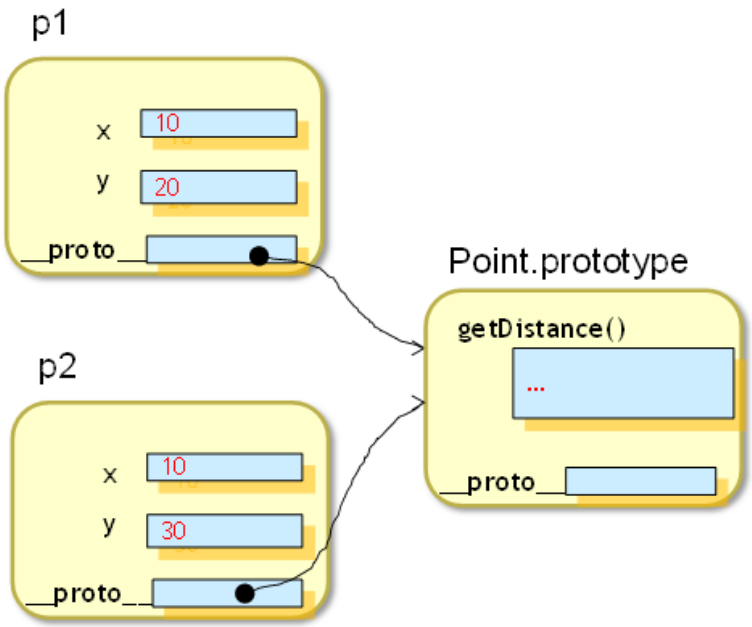
```
function Point(xpos, ypos) {
   this.x = xpos;
   this.y = ypos;
   this.getDistance = function () {
          return Math.sqrt(this.x * this.x + this.y * this.y);
   };
}
var p1 = new Point(10, 20);
var p2 = new Point(10, 30);
```



프로토타입

- 자바스크립트의 모든 객체들은 prototype이라는 숨겨진 객체를 가지고 있으며 이 객체를 이용하여 공유되는 메소드를 작성할 수 있다.

```
function Point(xpos, ypos) {
   this.x = xpos;
   this.y = ypos;
}

Point.prototype.getDistance = function ()
{
   return Math.sqrt(this.x * this.x + this.y
* this.y);
};
```



클래스, 프로토타입

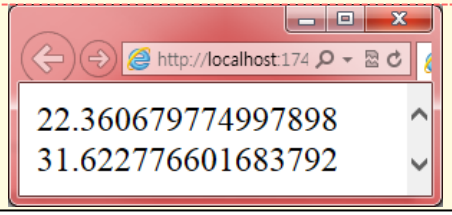
- 데이터는 객체의 전유물로 취급하는 것이 좋을 때가 많지만 메소드는 여러 객체가 공유하는 것이 좋을 때가 있다.
- 자바스크립트는 “클래스”를 지원하지 않기 때문에 “prototype”을 이용하여 여러 객체가 공유하는 메소드를 정의한다.
- “prototype 객체”는 여러 인스턴스를 포괄하는 상위 개념(일반적인 객체지향 언어에서는 “클래스”에 해당)이기 때문에 “prototype”을 사용하면 하나의 메소드 복사본을 이용하여 모든 인스턴스의 요구를 서비스할 수 있다 → “this.method”를 사용하면 인스턴스의 모든 복사본에 해당 method의 코드가 중복 복사된다.

프로토타입 예제

```
<!DOCTYPE html>
<html>
<body>
   <script>
      function Point(xpos, ypos) {
         this.x = xpos;
         this.y = ypos;
      }
      Point.prototype.getDistance = function ( ) {
         return Math.sqrt(this.x * this.x + this.y * this.y);
      }
      var p1 = new Point(10, 20);
      var d1 = p1.getDistance();
      var p2 = new Point(10, 30);
      var d2 = p2.getDistance();
      document.writeln(d1 + "<br />");
      document.writeln(d2 + "<br />");
   </script>
</body>
</html>
```

"Point"라는 객체의 "prototype"으로 "getDistance" 메소드를 추가한다(11번 슬라이드 내용 상기) → "Point" 객체는 모두 "getDistance()"를 공유할 수 있다.

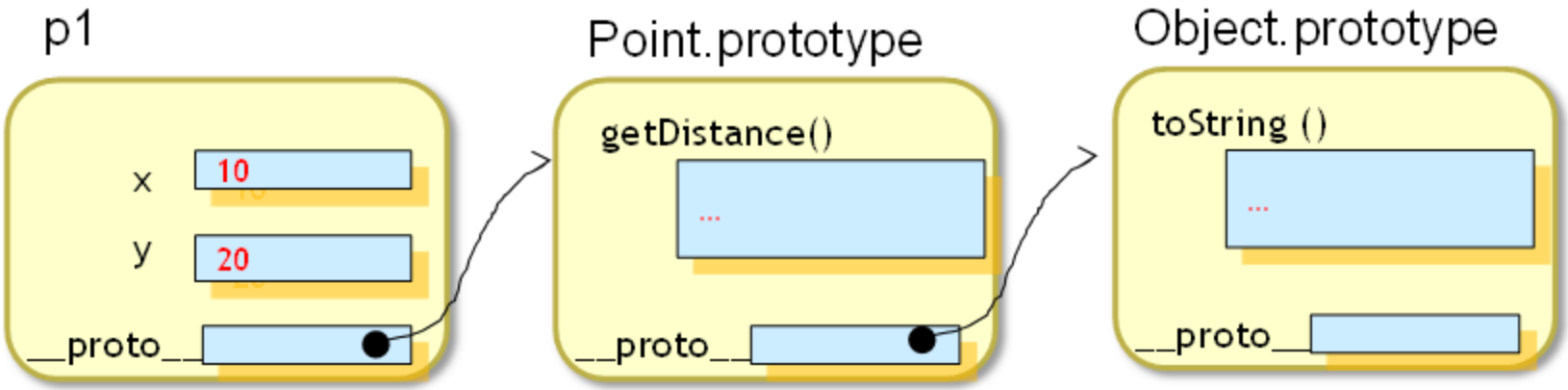
- getDistsance를 호출한 인스턴스 복사본("this"를 이용하여 접근)의 property x와 property y를 이용하여 계산(따라서 별도의 인자를 사용하지 않는다)
- Prototype method 호출방법은 일반적인 method 호출 방법과 동일



프로토타입 체인

- 자바스크립트에서 속성이나 메소드를 참조하게 되면 다음과 같은 순서대로 찾는다.

1. 객체 안에 속성이나 메소드가 정의되어 있는지 체크한다.
2. 객체 안에 정의되어 있지 않으면 객체의 prototype이 속성이나 메소드를 가지고 있는지 체크한다.
3. 원하는 속성/메소드를 찾을 때까지 프로토타입 체인(chain)을 따라서 올라간다.



Object 객체

- “Object”객체는 자바스크립트 객체의 부모가 되는 객체이다.
- 모든 객체는 내부에 Object 객체의 속성과 메소드를 가지고 있다.
- 자바스크립트는 Object, Math, Array, String과 같은 객체를 기본적으로 보유하고 있다.

자바 스크립트 내장 객체

- String 객체
- Date 객체
- Array 객체
- ...

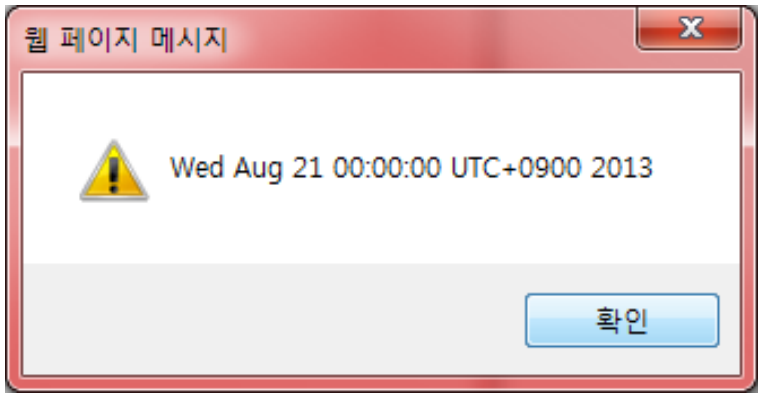
Date 객체

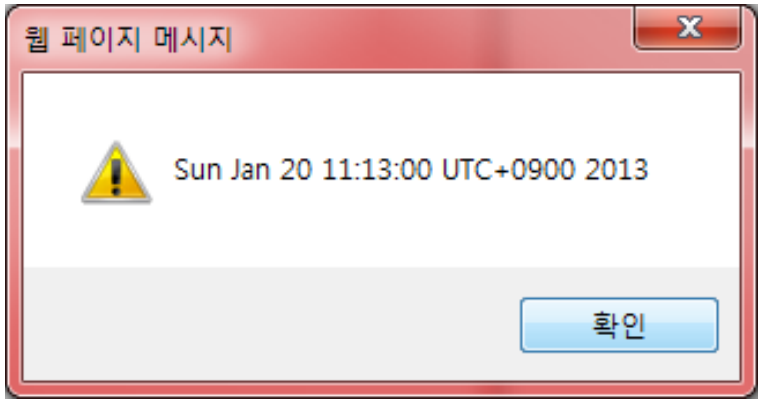
- Date 객체는 날짜와 시간 작업을 하는데 사용되는 가장 기본적인 객체
  - new Date() // 현재 날짜와 시간
  - new Date(milliseconds) //1970/01/01 이후의 밀리초
  - new Date(dateString)// 다양한 문자열
  - new Date(year, month, date[, hours[, minutes[, seconds[,ms]]]])

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>
```

1. 연, 월, 일, 시, 분, 초
2. JavaScript에서 월은 "0"부터 시작한다 → 즉, 예제는 2013년 8월 21일을 의미한다





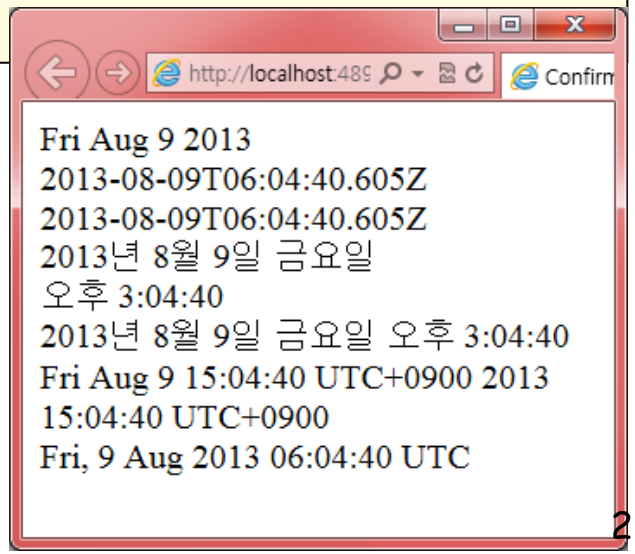
Date 객체의 메소드

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



UTC (국제 표준시)
- 프랑스어: Temps Universel Coordonné
- 영　　어: Coordinated Universal Time
- 그리니치 평균시(Greenwich Mean Time, GMT)와 유사

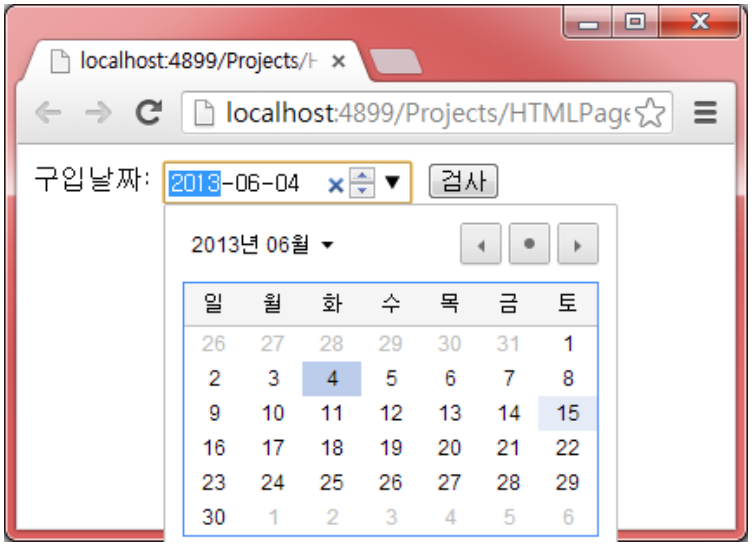
날짜 비교 예제

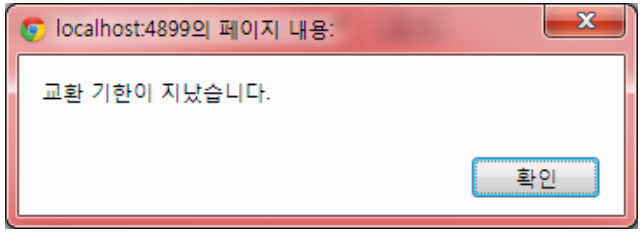
```
<!DOCTYPE html>
<html>
<head>
   <script>
      function checkDate() {
         var s = document.getElementById("pdate").value;
         var pdate = new Date(s);
         var today = new Date();
         var diff = today.getTime() - pdate.getTime();
         var days = Math.floor(diff / (1000 * 60 * 60 * 24));

         if (days > 30) {
            alert("교환 기한이 지났습니다.");
         }
      }
   </script>
</head>
```

The getTime() method returns the number of milliseconds between midnight of January 1, 1970 and the specified date.

예제

```
<body>
        구입날짜:
        <input type="date" id="pdate">
        <button onclick="checkDate()">검사</button>
</body>
</html>
```



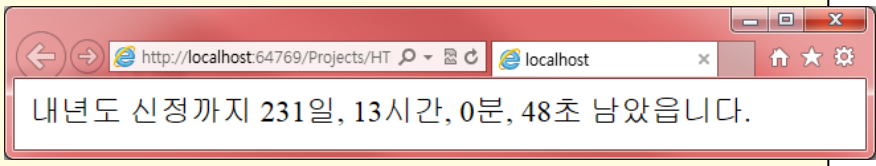


크롬에서 실행

타이머 예제

실행

```
<div id='remaining'></div>
<script>
   function datesUntilNewYear() {
      var now = new Date();
      var newYear = new Date('January 1, ' + (now.getFullYear() + 1));
      var diff = newYear - now;
      var milliseconds = Math.floor(diff % 1000);
      diff = diff / 1000;
      var seconds = Math.floor(diff % 60);
      diff = diff / 60;
      var minutes = Math.floor(diff % 60);
      diff = diff / 60;
      var hours = Math.floor(diff % 24);
      diff = diff / 24;
      var days = Math.floor(diff);

      var outStr = '내년도 신정까지 ' + days + '일, ' + hours + '시간, ' + minutes;
      outStr += '분, ' + seconds + '초' + ' 남았읍니다.';

      document.getElementById('remaining').innerHTML = outStr;
      // 1초가 지나면 다시 함수를 호출한다.
      setTimeout("datesUntilNewYear()", 1000);
   }
   // 타이머를 시작한다.
   datesUntilNewYear();
</script>
```

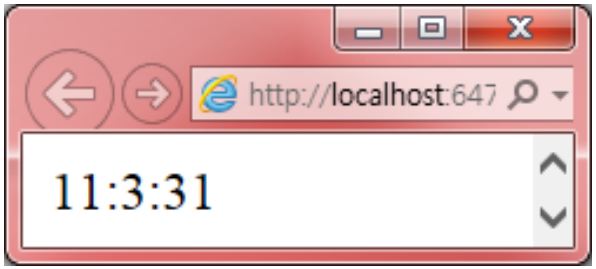


"innerHTML": HTML element에 접근할 수 있도록 도와주는 property이다.

시계 예제

```
<div id='clock'></div>

<script>
    function setClock() {
        var now = new Date();
        var s = now.getHours() + ':' + now.getMinutes() + ':' + now.getSeconds();
        document.getElementById('clock').innerHTML = s;
        setTimeout('setClock()', 1000);

    }
    setClock();
</script>
```

실행



Number 객체

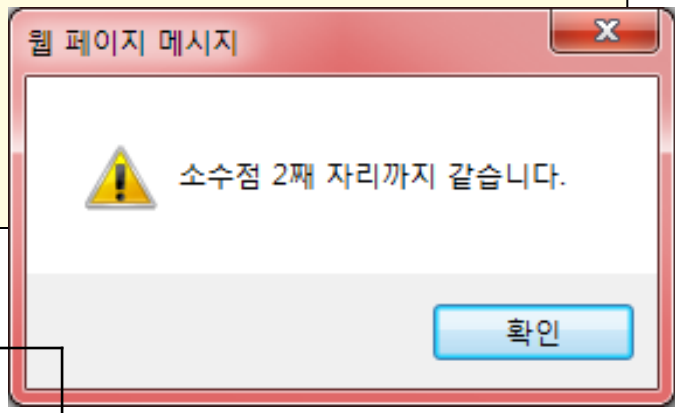
- Number 객체는 수치형 값을 감싸서 객체로 만들어 주는 랩퍼(wrapper) 객체
  - var num = new Number(7);

- 메소드
  - toFixed([digits])
    - var num = 123.456789;
    - document.writeln(num.toFixed(1) + '
'); // 123.5
  - toPrecision([precision))
    - var num = 123.456789;
    - document.writeln(num.toPrecision(1) + '
'); // 1e+2
  - toString([radix])

교재 351페이지

예제

```
<script>
    var count1, count2;
    count1 = new Number(1.237);
    count2 = 1.238;
    if (count1.toFixed(2) === count2.toFixed(2))
        alert("소수점 2째 자리까지 같습니다.");
</script>
```



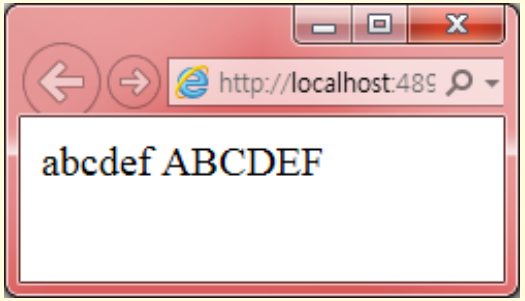
	x의 값이 5일 때		
연산자	설명	비교	결과
==	~ 동일한	x == 8	false
		x == 5	true
===	동일한 value와 type	x === "5"	false
		x === 5	true
!=	동일하지 않은	x ! = 8	true
!==	value또는 type이 동일하지 않은	x ! == "5"	true
		x ! == 5	false

String 객체

- 속성
  - length
  - prototype
  - constructor
- 메소드
  - charAt()
  - concat()
  - indexOf()
  - lastIndexOf()
  - match()
  - replace()
  - search()
  - slice()
  - ...

예제

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



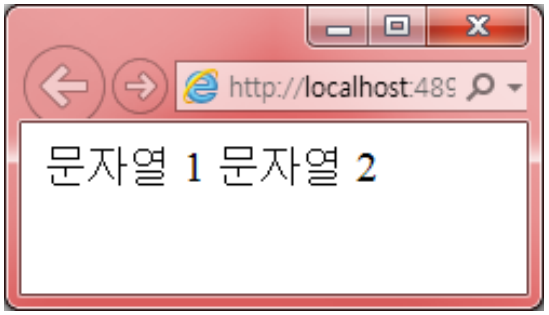
참고: writeln은 <pre>태그와 함께 사용 해야만 줄 바꿈이 된다

abcdef
ABCDEF

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

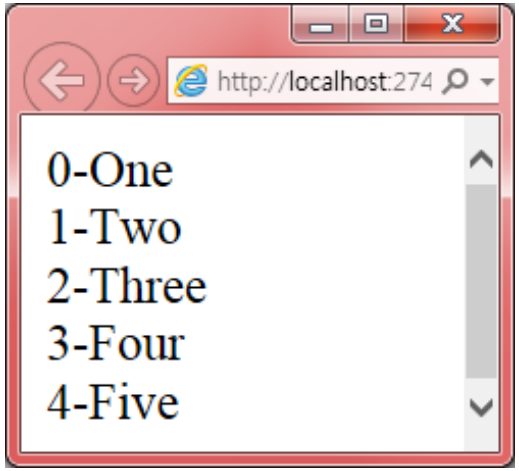
예 제

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

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



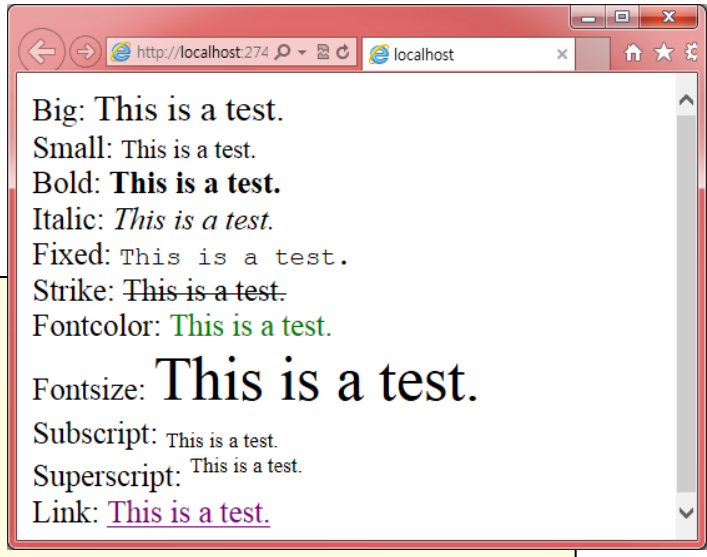
예제

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



예제

```
<script>
    var s = "This is a test.";
    document.write("Big: " + s.big() + "<br>");
    document.write("Small: " + s.small() + "<br>");
    document.write("Bold: " + s.bold() + "<br>");
    document.write("Italic: " + s.italics() + "<br>");
    document.write("Fixed: " + s.fixed() + "<br>");
    document.write("Strike: " + s.strike() + "<br>");
    document.write("Fontcolor: " + s.fontcolor("green") + "<br>");
    document.write("Fontsize: " + s.fontsize(6) + "<br>");
    document.write("Subscript: " + s.sub() + "<br>");
    document.write("Superscript: " + s.sup() + "<br>");
    document.write("Link: " + s.link("http://www.google.com") + "<br>");
</script>
```



Math 객체

속성	설명
E	오일러의 상수 (약 2.718)
LN2	자연 로그(밑수: 2) (약 0.693)
LN10	자연 로그(밑수:10) (approx. 2.302)
PI	파이 상수 (약 3.14)
SQRT1_2	1/2의 제곱근(약 0.707)
SQRT2	2의 제곱근 (약 1.414)

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 1 사이의 난수값 반환
round(x)	반올림
sqrt(x)	제곱근

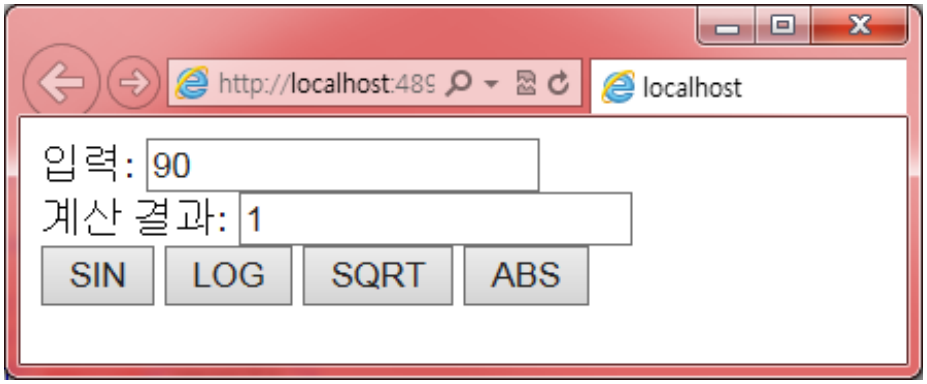
계산기 예제

```
<html>

<head>
   <script>
      function calc(type) {
         x = Number(document.calculator.number1.value);
         if (type == 1)
            y = Math.sin((x * Math.PI) / 180.0);
         else if (type == 2)
            y = Math.log(x);

         else if (type == 3)
            y = Math.sqrt(x);
         else if (type == 4)
            y = Math.abs(x);
         document.calculator.total.value = y;
      }
   </script>
</head>
```

chap9_str_math.html

예제

```
<body>
   <form name="calculator">
      입력:            <input type="text" name="number1"><br />
      계산 결과:        <input type="text" name="total"><br />
      <input type="button" value="SIN" onclick="calc(1);">
      <input type="button" value="LOG" onclick="calc(2);">
      <input type="button" value="SQRT" onclick="calc(3);">

      <input type="button" value="ABS" onclick="calc(4);">
   </form>

</body>
</html>
```



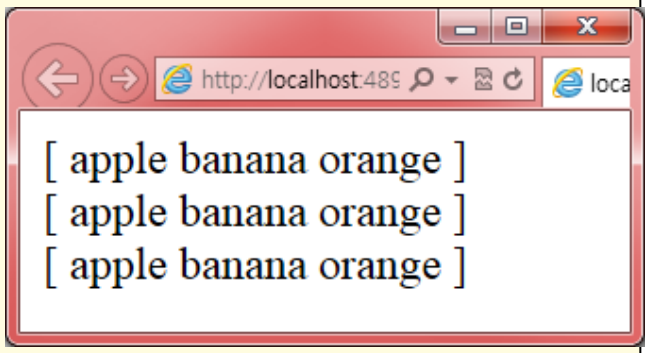
Array 객체

- 배열을 나타내는 객체

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

예제

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

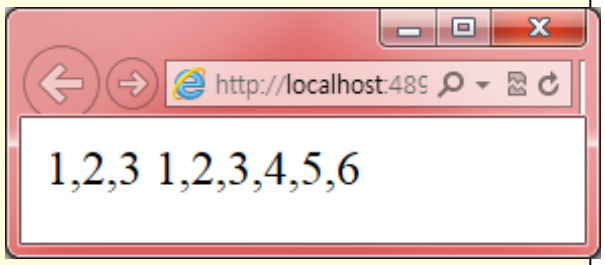


Array 객체의 메소드

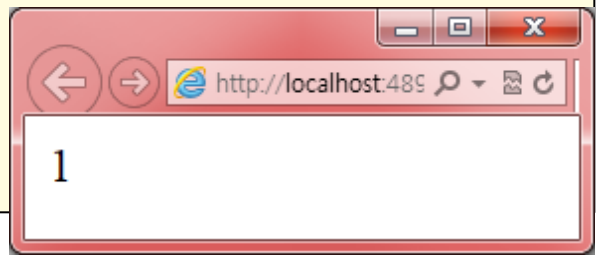
- 속성
  - length, prototype
- 메소드
  - concat()
  - indexOf()
  - join()
  - lastIndexOf()
  - pop()
  - push()
  - shift()
  - slice()
  - sort()
  - splice()

예제

```
<script>
    var x = [1, 2, 3];
    var y = [4, 5, 6];
    var joined = x.concat(y);

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



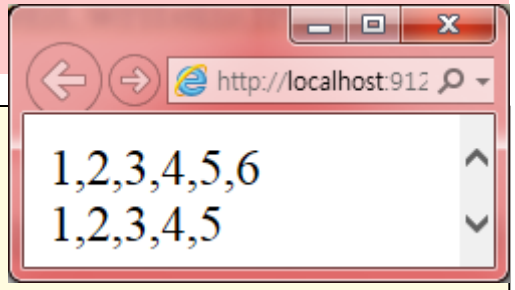

```
<script>
    var fruits = ["apple", "banana", "grape"];
    document.writeln(fruits.indexOf("banana"));
</script>
```



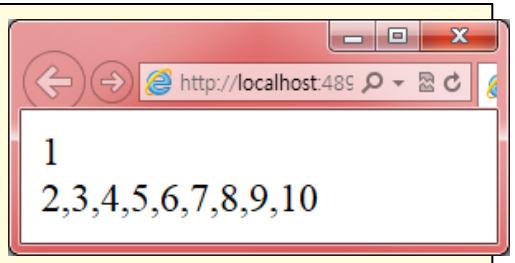
Chap9_Array.html

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>
```

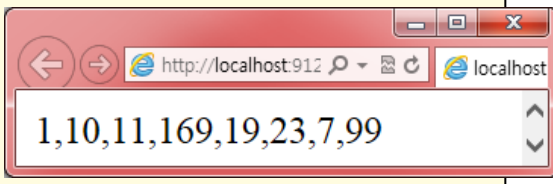



```
<script>
    var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

    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>
```

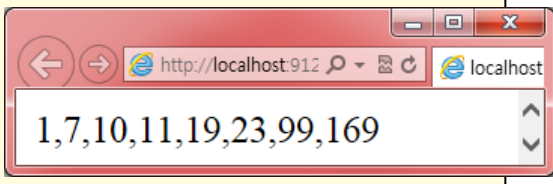


예제

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



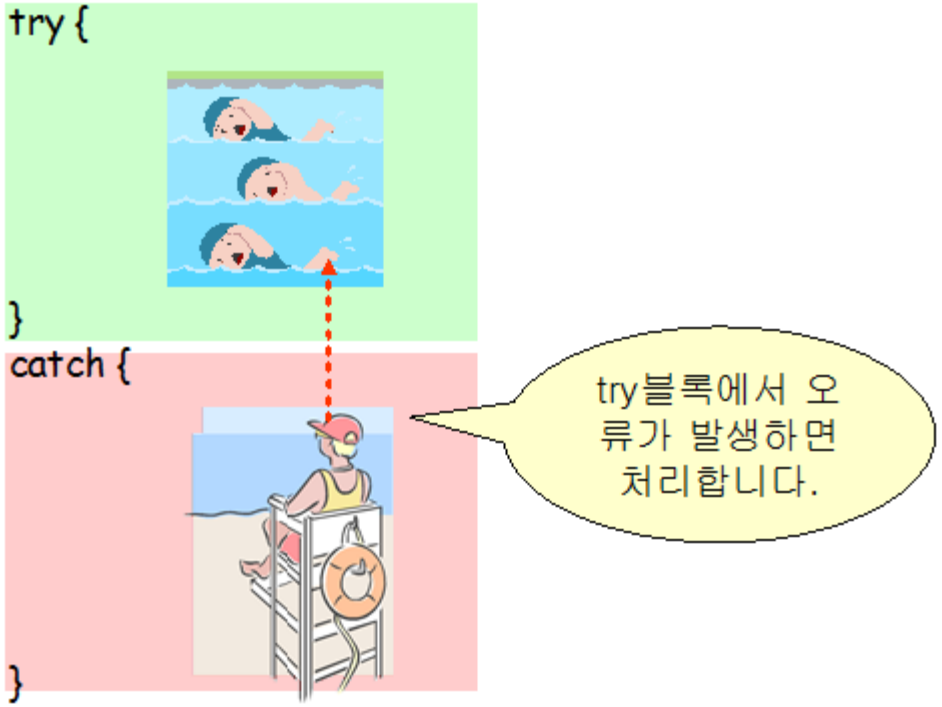

```
<script>
    var myArray = [10, 7, 23, 99, 169, 19, 11, 1];
    myArray.sort(function (a, b) { return a - b });
    document.writeln(myArray);
</script>
```



오류 처리

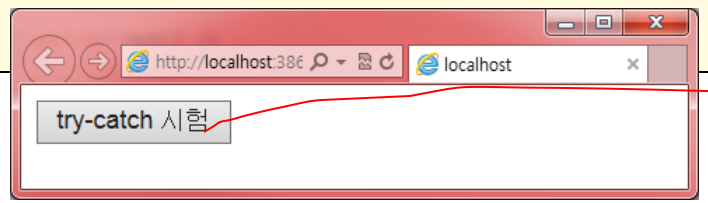
- 자바스크립트에서의 예외 처리기는 try 블록과 catch 블록으로 이루어진다.

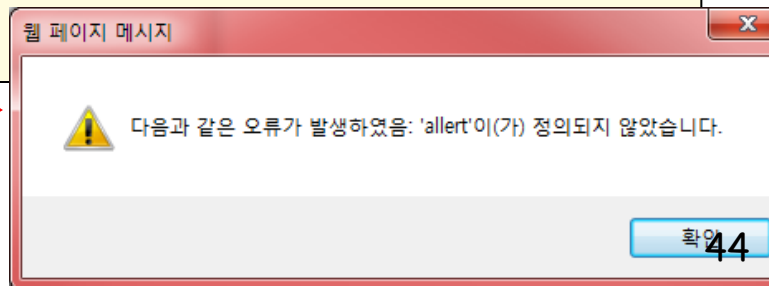
```
try
{
        // 예외가 발생할 수 있는 코드
}

catch (변수)
{
        // 예외를 처리하는 코드
}
```



예제

```
<!DOCTYPE html>
<html>
<head>
    <script>
        var msg = "";
        function test() {
            try {
                allert("Hello World!");
            }
            catch (error) {
                msg = "다음과 같은 오류가 발생하였음: " + error.message;
                alert(msg);
            }
        }
    </script>
</head>
<body>
    <input type="button" value="try-catch 시험" onclick="test()" />
</body>
</html>
```





throw 문장

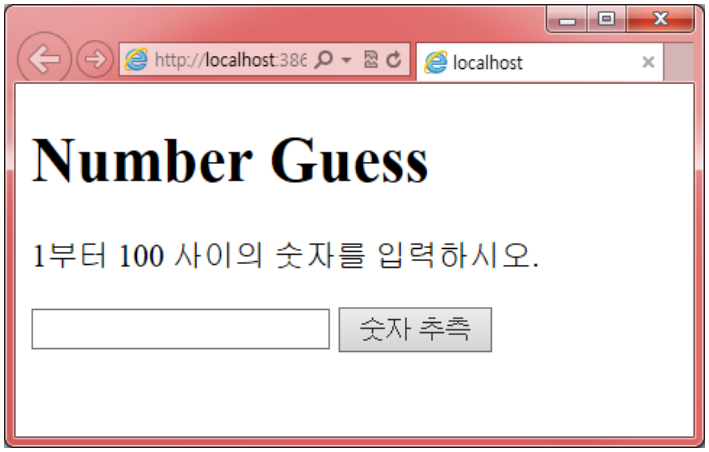
- throw 문장은 개발자가 오류를 생성할 수 있도록 한다.
- throw 문장을 사용하여서 오류 처리를 이용할 수도 있다.
  - (예) 숫자 맞추기 게임

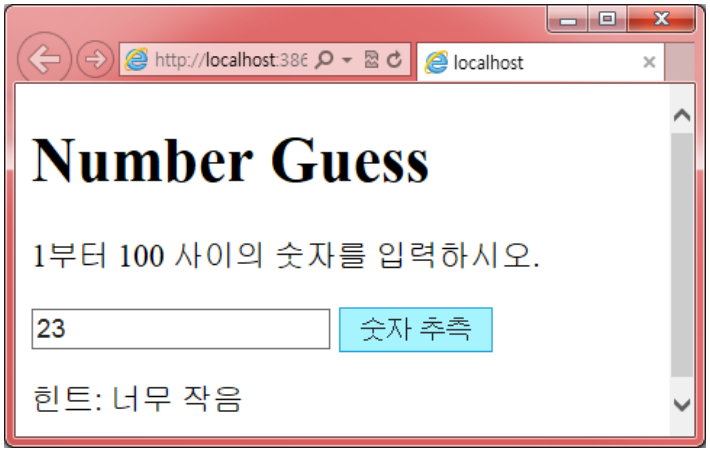
예제

```
<!DOCTYPE html>
<html>
<body>
    <script>
        var solution = 53;
        function test() {
            try {
                var x = document.getElementById("number").value;
                if (x == "") throw "입력없음";
                if (isNaN(x)) throw "숫자가 아님";
                if (x > solution) throw "너무 큼";
                if (x < solution) throw "너무 작음";
                if (x == solution) throw "성공";
            }
            catch (error) {
                var y = document.getElementById("message");
                y.innerHTML = "힌트: " + error;
            }
        }
    </script>
```

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





Q & A