



CHAPTER 9.

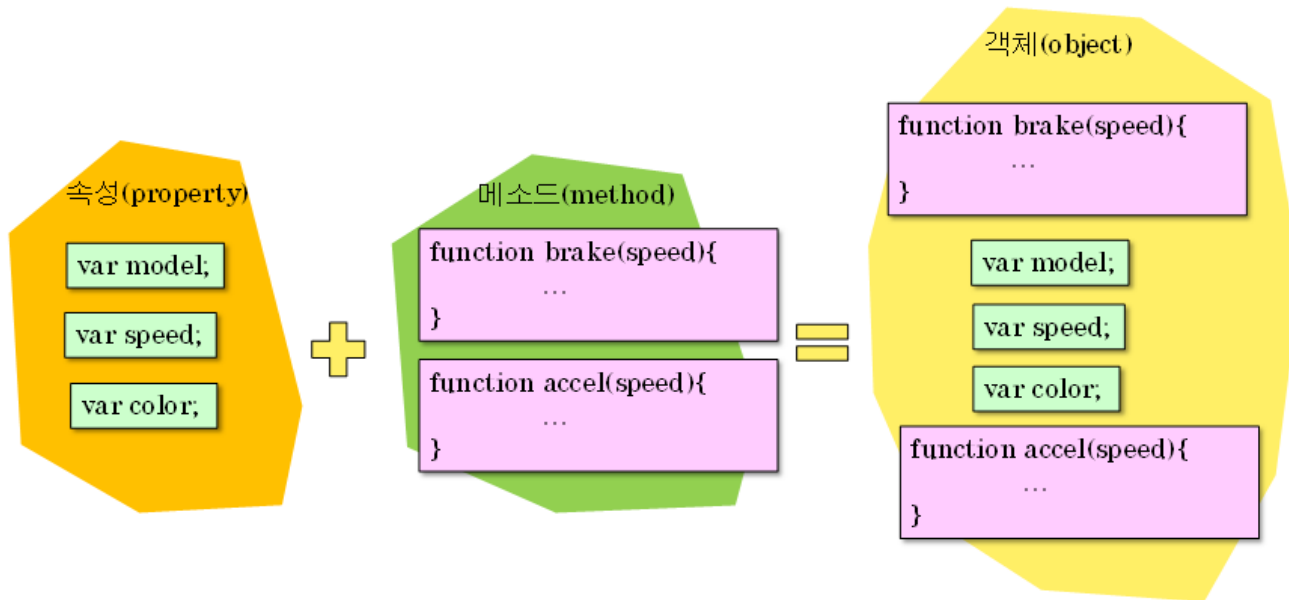
자바 스크립트 객체





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.





객체의 종류

- 객체의 2가지 종류
 - *내장 객체(built-in object)*: 생성자가 미리 작성되어 있다.
 - *사용자 정의 객체(custom object)*: 사용자가 생성자를 정의한다.
- 내장 객체들은 생성자를 정의하지 않고도 사용이 가능하다. `Date`, `String`, `Array`와 같은 객체들이 내장 객체이다.



객체 생성 방법

- 객체를 생성하는 2가지 방법
 - 객체를 객체 상수로부터 직접 생성한다.
 - 생성자 함수를 이용하여 객체를 정의하고 `new`를 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; }  
};
```

객체의 속성

객체의 메소드

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



생성자를 이용한 객체 생성

생성자도 함수
이다.

생성자 이름은 항상 대문자로 한다.

```
function Car(model, speed, color) {
```

this 키워드로
일반 변수와
객체 속성을
구별한다.

```
this.model = model;
```

```
this.speed = speed;
```

```
this.color = color;
```

객체의 속성

```
this.brake = function () {
```

```
  this.speed -= 10;
```

```
}
```

```
this.accel = function () {
```

```
  this.speed += 10;
```

```
}
```

```
}
```

객체의 메소드



생성자를 이용한 객체 생성

이 변수에
객체가 저장
된다.

new 연산자는 새로운 객체를
만든다.

이 인수들
로 객체가
초기화된다.

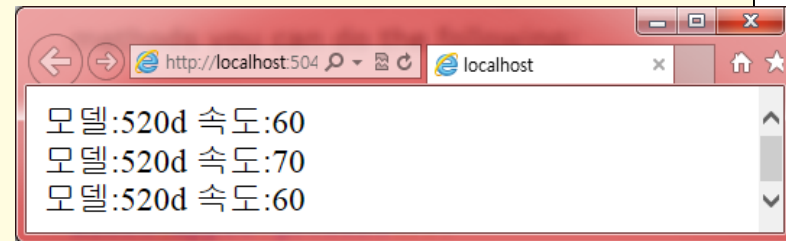
```
var myCar = new Car("520d", 60, "white");  
myCar.color = "yellow";  
myCar.brake();
```



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.

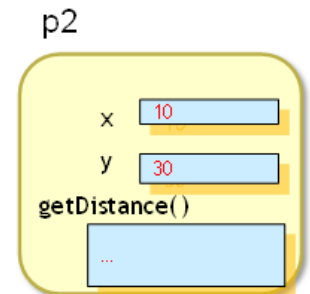
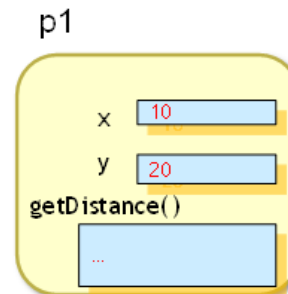
```
myCar.turbo = true;  
myCar.showModel = function() {  
    alert( "모델은 " + this.model + "입니다." )  
}
```



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);
```



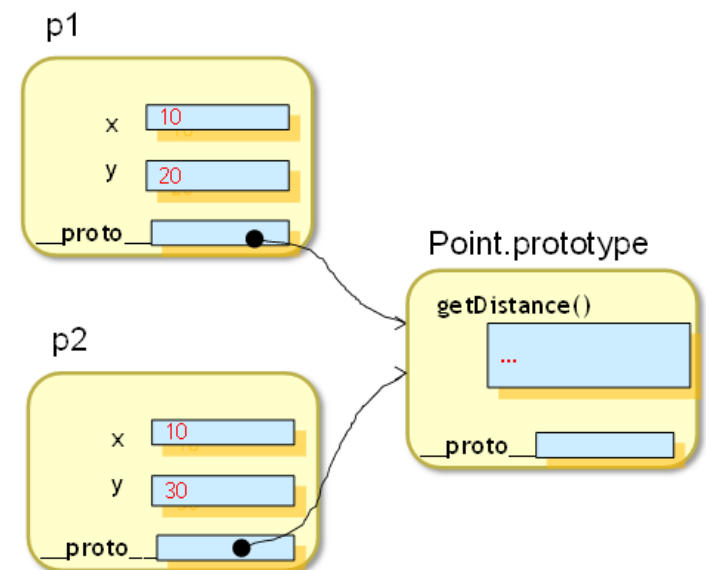


프로토타입

- 자바스크립트의 모든 객체들은 prototype이라는 숨겨진 객체를 가지고 있으며 이 객체를 이용하여서 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);  
};
```

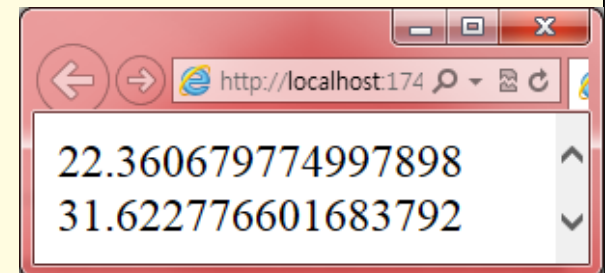




프로토타입 예제

```
<!DOCTYPE html>
<html>
<body>
  <script>
    function Point(xpos, ypos) {
      this.x = xpos;
      this.y = ypos;
    }
    Point.prototype.getDistance = function (p) {
      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 />");

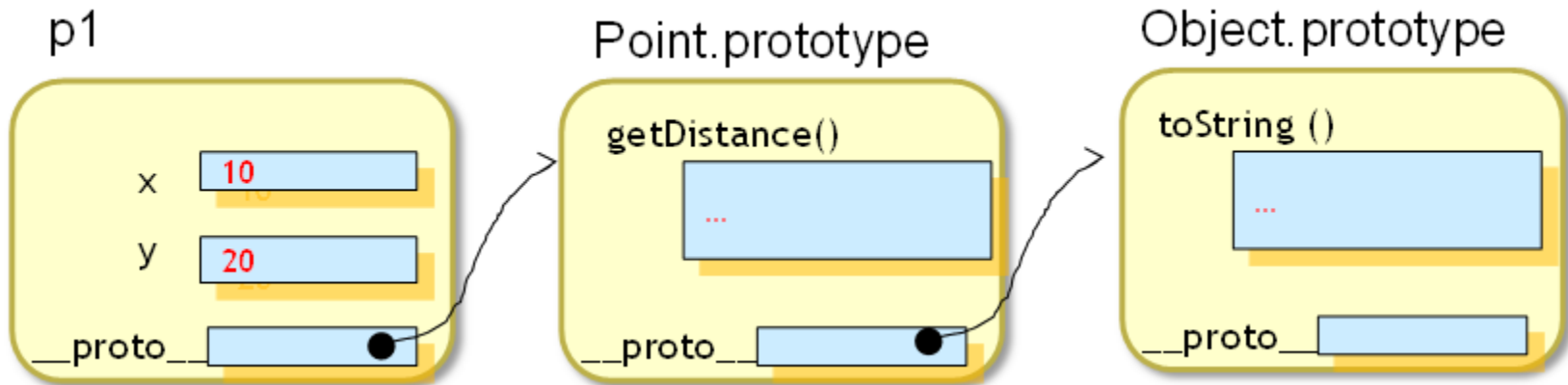
  </script>
</body>
</html>
```





프로토타입 체인

- 자바스크립트에서 속성이나 메소드를 참조하게 되면 다음과 같은 순서대로 찾는다.
 - 객체 안에 속성이나 메소드가 정의되어 있는지 체크한다.
 - 객체 안에 정의되어 있지 않으면 객체의 `prototype`이 속성이나 메소드를 가지고 있는지 체크한다.
 - 원하는 속성/메소드를 찾을 때까지 프로토타입 체인(chain)을 따라서 올라간다.





자바 스크립트 내장 객체

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



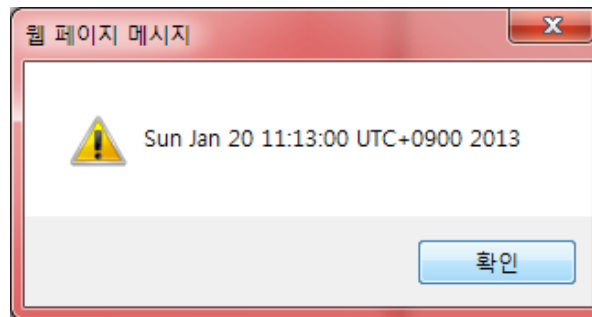
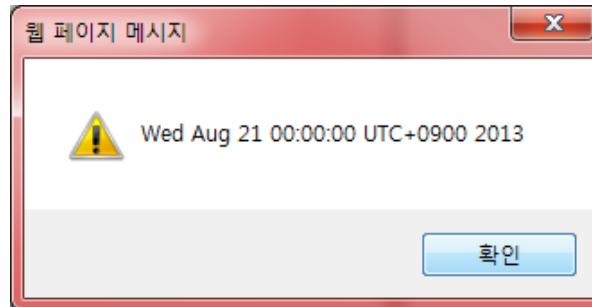
Date 객체

- 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>
```





Date 객체의 메소드

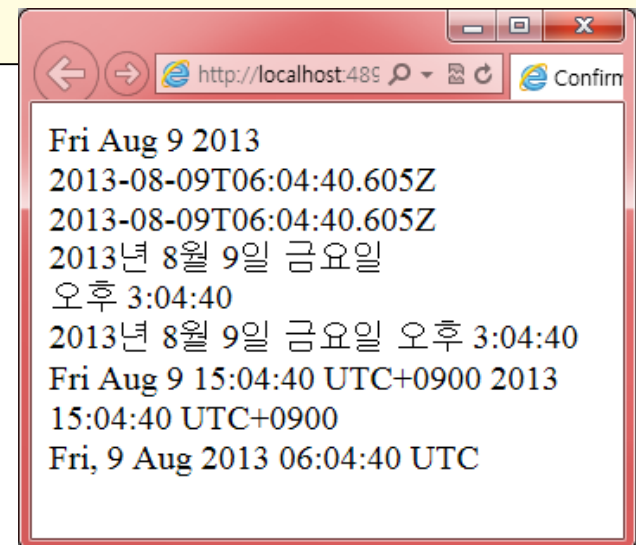
- getDate() (1-31 반환)
- getDay() (0-6 반환) index 수를 의미
- getFullYear() (4개의 숫자로 된 연도 반환)
- getHours() (0-23 반환)
- getMilliseconds()(0-999)
- getMinutes()(0-59)
- getMonth()(0-11)
- getSeconds()(0-59)

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



예제

```
<script>
  var today = new Date();
  document.write(today.toString() + "<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.getTimeString() + "<br>");
  document.write(today.toUTCString() + "<br>");
</script>
```





날짜 비교 예제

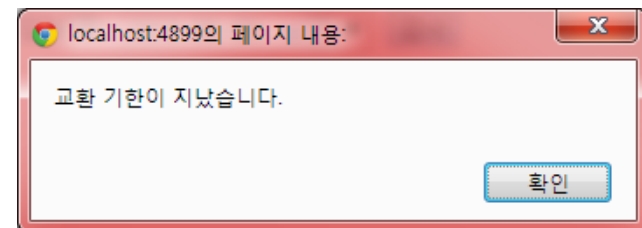
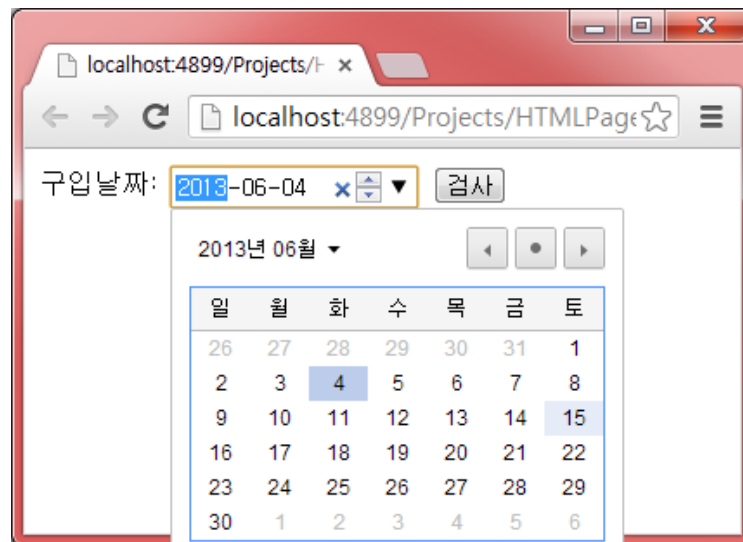
```
<!DOCTYPE html>
<html>
<head>
  <script>
    function checkDate() {
      var s = document.getElementById("pdate").value;
      var pdate = new Date(s);
      var today = new Date();
      var diff = today.getTime() - pdate.getTime();
      var days = Math.floor(diff / (1000 * 60 * 60 * 24));

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



예제

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



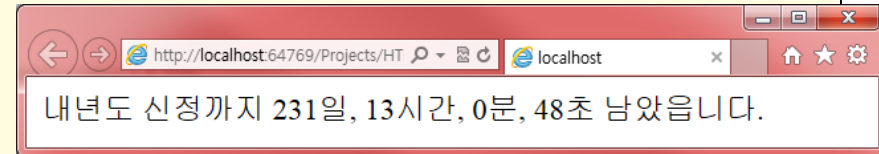


타이머 예제

```
<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' + ' 남았습니다.';

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





시계 예제

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

```
<script>
```

```
function setClock() {
```

```
    var now = new Date();
```

```
    var s = now.getHours() + ':' + now.getMinutes() + ':' + now.getSeconds();
```

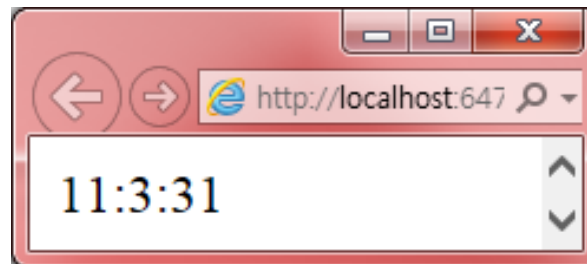
```
    document.getElementById('clock').innerHTML = s;
```

```
    setTimeout('setClock()', 1000);
```

```
}
```

```
    setClock();
```

```
</script>
```





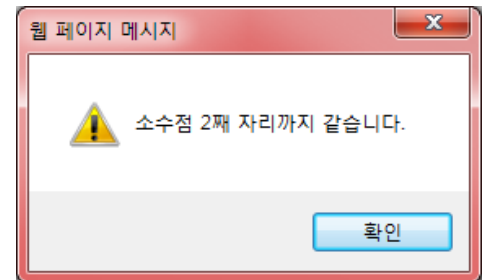
Number 객체

- Number 객체는 수치형 값을 감싸서 객체로 만들어 주는 래퍼 (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])`



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>
```





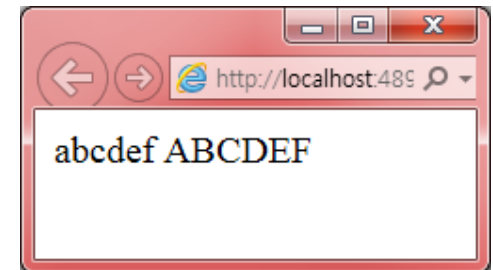
String 객체

- 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>
```

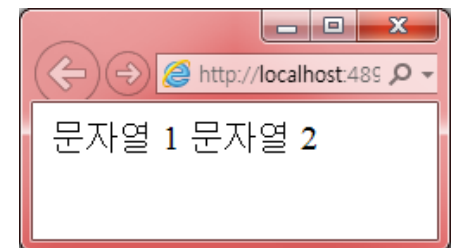




예제

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

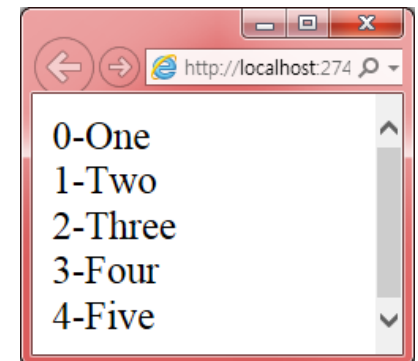
  s3 = s1.concat(s2);
  document.writeln(s3 + '<br>'); // “문자열 1 문자열 2“
</script>
```





예제

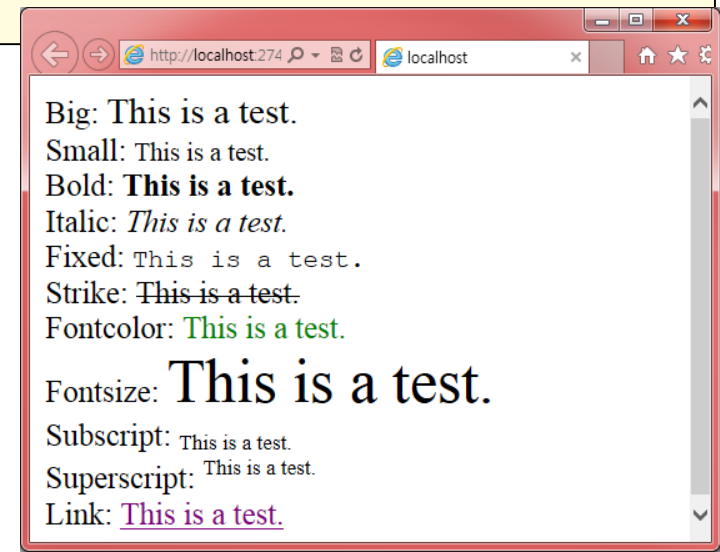
```
<script>
  s = "One,Two,Three,Four,Five";
  array = s.split(',');
  for (i = 0; i < array.length; i++) {
    document.writeln(i + '-' + array[i] + '<BR>');
  }
</script>
```





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

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

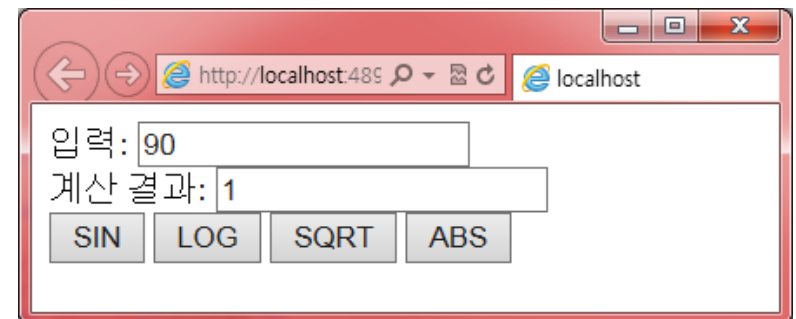


예제

```
<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";
```

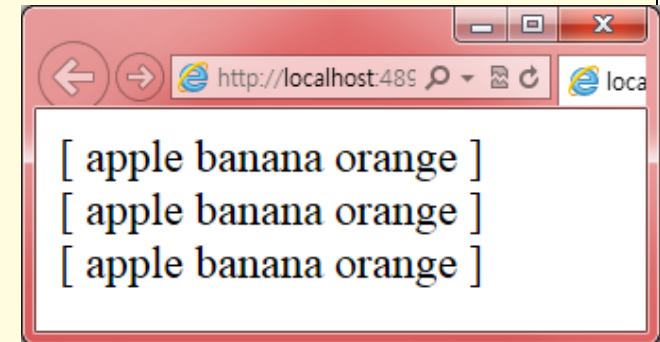


예제

```
<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

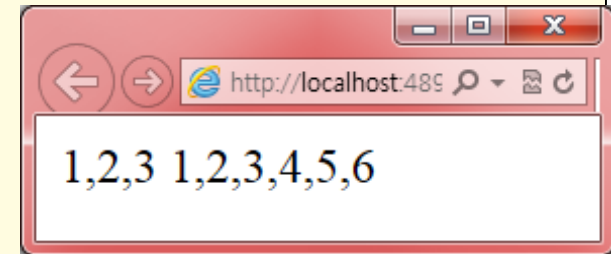
- 속성
 - length, prototype
- 메소드
 - concat()
 - indexOf()
 - join()
 - lastIndexOf()
 - pop()
 - push()
 - shift()
 - slice()
 - sort()
 - splice()



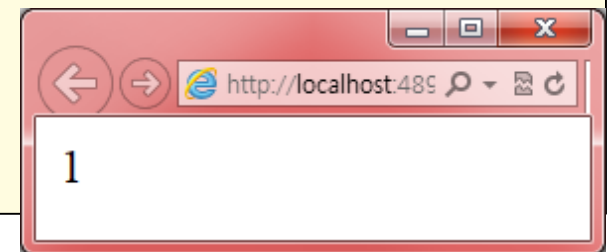
예제

```
<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"));    // 출력: 1
</script>
```

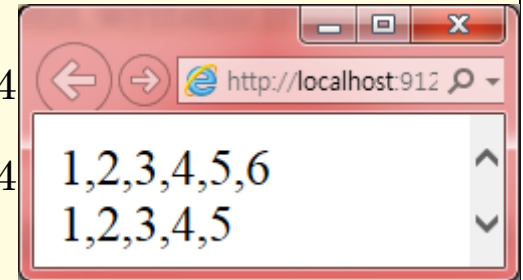




예제

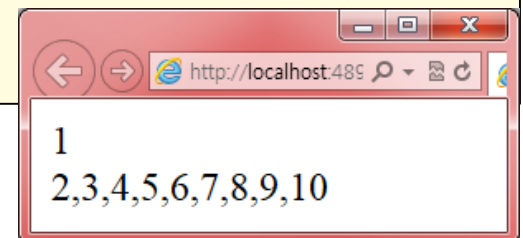
```
<script>
  var numbers = [1, 2, 3, 4, 5];

  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];

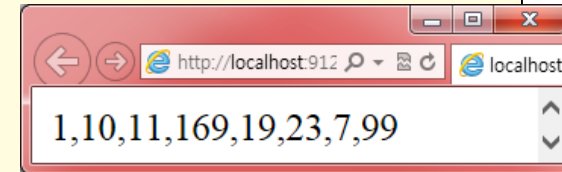
  var item = numbers.shift();
  document.writeln(item + '<BR>');        // 출력: 1
  document.writeln(numbers + '<BR>');    // 출력: 2,3,4,5,6,7,8,9,10
</script>
```



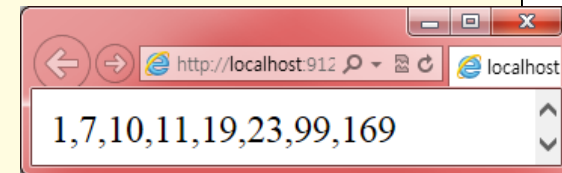


예제

```
<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>
```



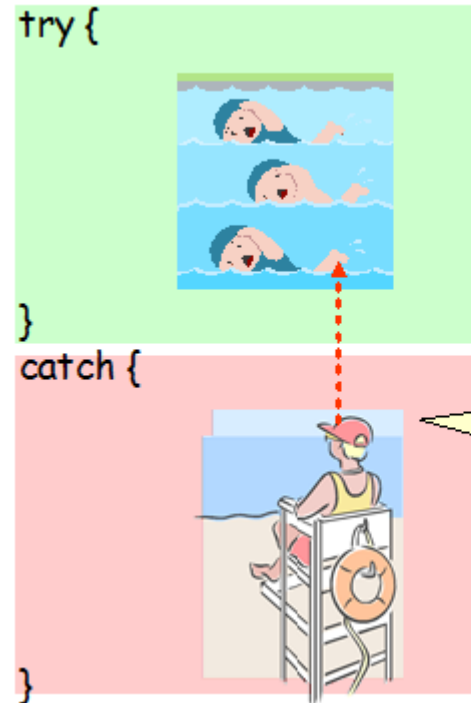


오류 처리

- 자바스크립트에서의 예외 처리기는 try 블록과 catch 블록으로 이루어진다.

```
try
{
    // 예외가 발생할 수 있는 코드
}

catch (변수)
{
    // 예외를 처리하는 코드
}
```

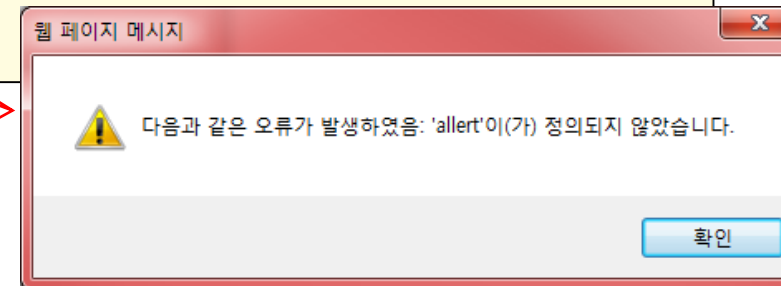
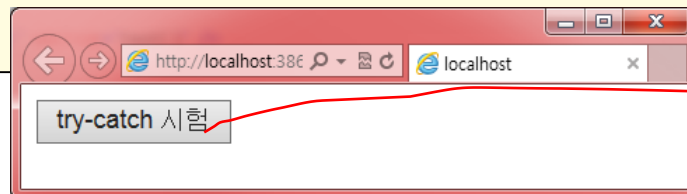


try블록에서 오류가 발생하면 처리합니다.



예제

```
<!DOCTYPE html>
<html>
<head>
  <script>
    var msg = "";
    function test() {
      try {
        alert("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 사용하여 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>
```

Number Guess

1부터 100 사이의 숫자를 입력하시오.

Number Guess

1부터 100 사이의 숫자를 입력하시오.

힌트: 너무 작음



Q & A

