웹 앱 개발을 위한 JavaScript 기초 강의 노트

제 11회차

내장 객체 - Object, Number, String

▣ 학습목표

- 내장 객체를 이해하고, 기본 자료형과 객체의 차이점을 설명할 수 있다.
- Object 객체의 속성과 메서드를 활용할 수 있다.
- Number 객체의 속성과 메서드를 활용할 수 있다.
- String 객체의 속성과 메서드를 활용할 수 있다.

▣ 학습내용

- 내장객체의 개요
- 기본 내장 객체의 종류

1. 내장객체의 개요

1) 내장객체란?

- 객체 생성 방법

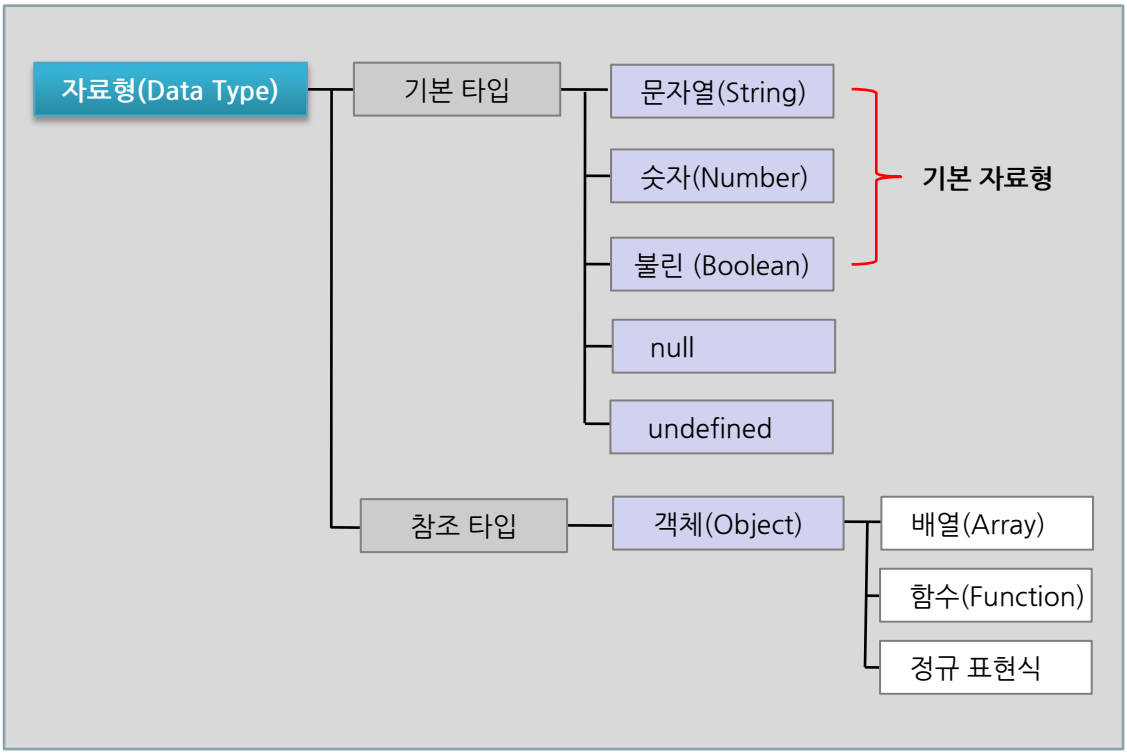
- 중괄호 안에 속성과 메서드를 포함하기
- 생성자 함수 사용하기

- 내장객체

- 내장함수처럼 JavaScript 자체에서 제공하는 객체
- 다양한 내장 개체 존재
- 각 객체마다 다양한 속성과 메서드 제공

2) 기본 자료형과 객체의 차이

- 자료형의 분류



1. 내장객체의 개요

2) 기본 자료형과 객체의 차이

- 기본 자료형 vs 객체

객체	기본 자료형
• 속성, 메서드를 가짐	• 속성, 메서드 없음 • 속성과 메서드를 사용하면 자동으로 객체로 변환됨 • 문자열 → String 객체 • 숫자 → Number 객체 • 불린 → Boolean 객체

- 예 : 기본 자료형인 number1과 객체인 number2에 print() 메서드 추가 시 결과의 차이

```
<script>
    var number1 = 123;
    var number2 = new Number(123);

    var out = '';
    out += 'number1의 type :  ' + typeof(number1) +'₩t 값 : '+ number1 +
'₩n₩n';
    out += 'number2의 type :  ' + typeof (number2) + '₩t 값 : ' + number1 +
'₩n';
    alert(out);

    number1.print = function () {
       alert('number1');
    }

    number2.print = function () {
       alert('number2');
    }

    number2.print();
    number1.print();

</script>
```

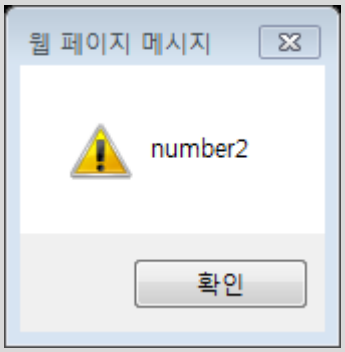
1. 내장객체의 개요

2) 기본 자료형과 객체의 차이

- 기본 자료형 vs 객체

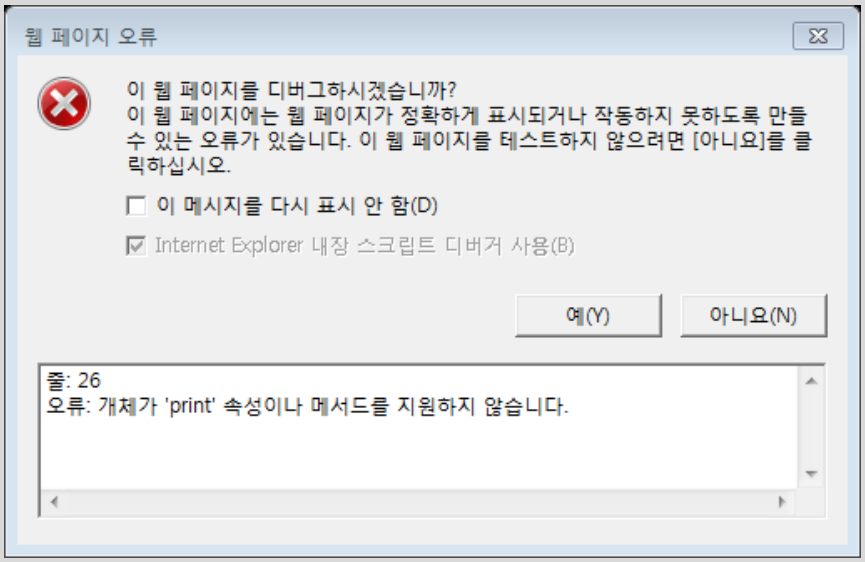
• 결과

① number2.print();



→ number2는 객체이기 때문에 생성 후 메서드를 추가하여 사용 가능

② number1.print();



→ number1은 기본 자료형이기 때문에 속성과 메서드를 사용할 때는 일시적으로 객체로 변환되어 객체의 프로토타입을 공유하지만, 기본적으로 기본 자료형이기 때문에 속성과 메서드를 추가할 수는 없기 때문에 오류가 발생함

2. 기본 내장 객체의 종류

1) Object 객체

- Object 객체의 특징

- JavaScript의 가장 기본적인 내장 객체이자, 최상위 객체
- Object() 생성자 함수에 의해 생성된 인스턴스
- 모든 기본 내장 객체는 Object 객체를 기본으로 만들어짐
- 생성
  - 객체 표기법 (var object = { };)
  - 생성자 함수 (var object = new Object();)
- 메서드

메서드	설명
constructor()	객체의 생성자 함수를 나타냄
hasOwnProperty(V)	문자열 매개 변수로 전달된 속성이 객체에 있는지 확인
isPrototypeOf(object)	객체가 object의 프로토타입인지 검사
propertyIsEnumerable(V)	전달받은 속성이 반복문으로 열거할 수 있는지 확인
toLocaleString()	객체를 호스트 환경에 맞는 언어의 문자열로 변경
toString()	객체를 문자열로 변경
valueOf()	객체의 값을 나타냄

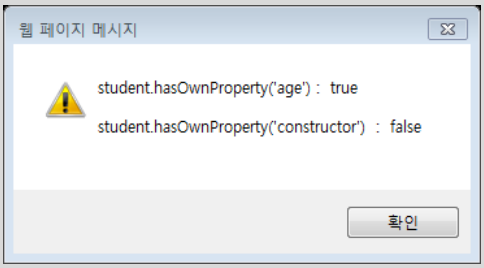
2. 기본 내장 객체의 종류

1) Object 객체

- hasOwnProperty()메서드와 propertyIsEnumerable() 메서드 출력

```
<script>
    var student = {                                    ①
        name: '홍길동',
        age: 20
    };

    var out = '';
    out += "student.hasOwnProperty('age') : " +                                    ②
student.hasOwnProperty('age')+'₩n₩n';
    out += "student.hasOwnProperty('constructor') : "+
student.hasOwnProperty('constructor')+'₩n₩n';
    alert(out);

    out = "student.propertyIsEnumerable('name') : " +
student.propertyIsEnumerable('name') + '₩n₩n';
    alert(out);

    student.toString = function () {
        return '재선언';
    }
   alert(student);
</script>
```

① name, age 속성 가지는 student 객체 생성
② age, constructor 속성 있는지 확인
③ 출력화면



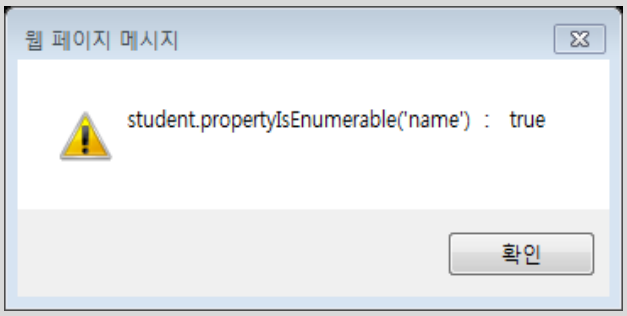
2. 기본 내장 객체의 종류

1) Object 객체

- hasOwnProperty()메서드와 propertyIsEnumerable() 메서드 출력

```
<script>
    var student = {
      name: '홍길동',
      age: 20
    };

    var out = '';
    out += "student.hasOwnProperty('age') : " +
student.hasOwnProperty('age')+'₩n₩n';
    out += "student.hasOwnProperty('constructor') : "+
student.hasOwnProperty('constructor')+'₩n₩n';
    alert(out);

  ④ out = "student.propertyIsEnumerable('name') : " +
student.propertyIsEnumerable('name') + '₩n₩n';
    alert(out);

    student.toString = function () {
       return '재선언';
    }
   alert(student);
</script>
```

④ name 속성을 반복문으로 열거할 수 있는지 확인
⑤ 출력 화면



2. 기본 내장 객체의 종류

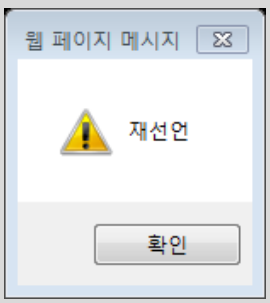
1) Object 객체

- toString() 메서드의 재선언

```
<script>
    var student = {
        name: '홍길동',
        age: 20
    };

    var out = '';
    out += "student.hasOwnProperty('age') : " +
student.hasOwnProperty('age')+'₩n₩n';
    out += "student.hasOwnProperty('constructor') : "+
student.hasOwnProperty('constructor')+'₩n₩n';
    alert(out);

    out = "student.propertyIsEnumerable('name') : " +
student.propertyIsEnumerable('name') + '₩n₩n';
    alert(out);

⑥  student.toString = function () {
        return '재선언';
    }
    alert(student);
</script>
```

⑥ student 객체의 toString 메서드의 값을 수정하면 재선언이라는 값을 돌려주도록 설정

⑦ 출력 화면



2. 기본 내장 객체의 종류

1) Object 객체

- constructor() 메서드를 이용한 자료형 비교
- 최상위 객체이기 때문에, 프로토타입에 속성이나 메서드를 추가하면 모든 객체에서 활용 가능

```
<script>
① var number1 = 123;
  var number2 = new Number(123);

  var out='';
② out='number1 : '+number1+'₩n'+'number2 : '+number2;
  alert(out);

③ if (typeof (number1) == typeof (number2)) {
    alert('typeof (number1) == typeof (number2)');
  }

④ if (number1.constructor == number2.constructor) {
    alert('number1.constructor == number2.constructor');
  }

⑤ Object.prototype.print = function () {
    alert('모든 객체에 추가');
  }

⑥ out.print();
  number2.print();

</script>
```

① 숫자로 선언된 number1을 선언하고, Number 생성자 함수를 이용해 number2 객체 생성

② number1과 number2를 출력 시 둘 다 동일하게 123을 출력함

③ typeof 연산자로 자료형을 확인하면 자료형이 달라 if 조건문의 조건이 거짓이 돼 출력 안함

④ number1과 number2의 생성자 함수를 비교하면 모두 Number 생성자 함수에 의해 생성되었기 때문에 if 조건문의 조건이 true이므로 대화상자 출력함.

⑤ Object객체의 프로토타입에 대화상자를 출력하는 print 메서드를 추가하면

⑥ out에서 print 메서드를 출력하면 out은 일시적으로 String 객체로 변환되어 print 메서드를 출력하게 되며, Number 객체인 number2에서도 print 메서드를 출력하여 대화상자 출력함

2. 기본 내장 객체의 종류

2) Number 객체

- Object 객체의 특징

- 숫자를 표현할 때 사용하는 객체
- 생성

  - 변수 선언 (var number1 = 123;)

  - 생성자 함수 사용 (var number2 = new Number(123);)
- 메서드

  - Object 객체의 7가지 메서드를 모두 가짐

  - 추가로 가지는 메서드

메서드	설명
toExponential()	숫자를 지수 표기법으로 변환한 문자열 리턴
toFixed(x)	숫자를 고정 소수점 표시로 변환한 문자열 리턴 (매개 변수로 입력된 x 만큼 소수점 자리를 나타냄)
toPrecision (x)	숫자를 길이에 따라 지수 표기법 혹은 고정 소수점 표시로 변환한 문자열 리턴 (매개 변수로 입력된 x의 길이로 숫자를 나타냄)

2. 기본 내장 객체의 종류

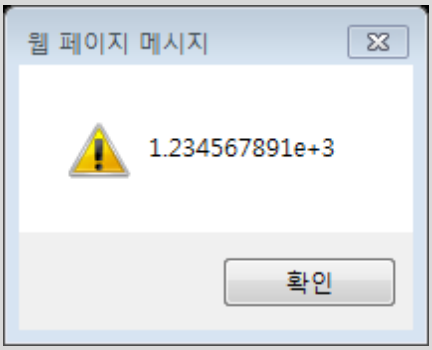
2) Number 객체

- Object 객체의 특징

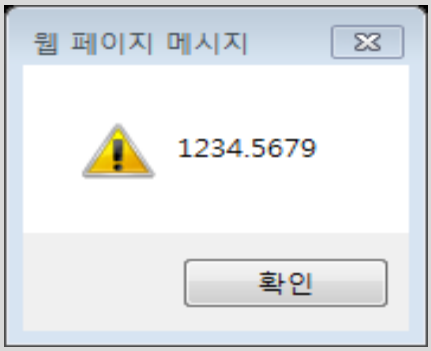
• 예

```
<script>
    var number1 = new Number(1234.567891);

 ① alert(number1.toExponential());
 ② alert(number1.toFixed(4));
 ③ alert(number1.toPrecision(5));
 ④ alert((1235.5674).toFixed(3));
 </script>
```

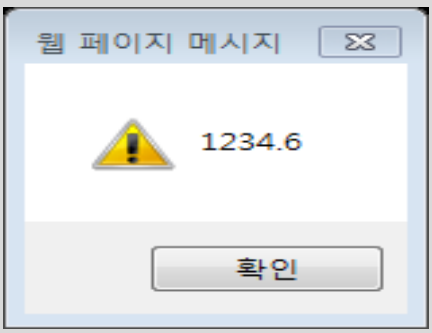
① 출력 화면



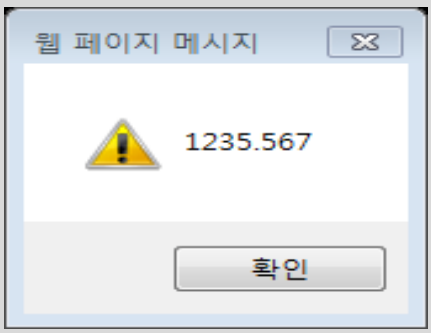
② 출력 화면



③ 출력 화면



④ 출력 화면



2. 기본 내장 객체의 종류

2) Number 객체

- Object 객체의 특징

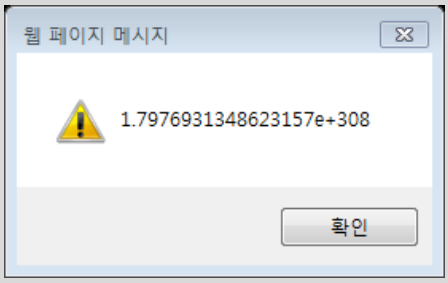
• Number 속성(Number 생성자 함수 속성)

메서드	설명
MAX_VALUE	JavaScript에서 나타낼 수 있는 가장 큰 숫자
MIN_VALUE	JavaScript에서 나타낼 수 있는 가장 작은 숫자
NEGATIVE_INFINITY	음의 무한대 숫자
NaN	숫자로 나타낼 수 없는 값
POSITIVE_INFINITY	양의 무한대 숫자
prototype	모든 Number 객체에 포함할 속성이나 메서드 추가 허용

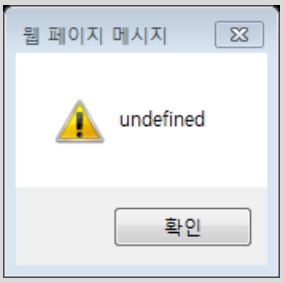
• 예

```
<script>
    var number1 = new Number(123);
① alert(Number.MAX_VALUE);
② alert(number1.MAX_VALUE);

    Number.prototype.test = function () {
       return this.valueOf() * 2;
    }

③ alert(number1.test());
</script>
```

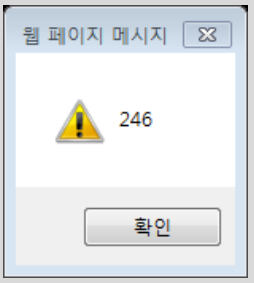
① 출력 화면



② 출력 화면



③ 출력 화면



2. 기본 내장 객체의 종류

3) String 객체

- String 객체의 특징

- 문자를 표현할 때 사용하는 객체
- 생성

  - 변수 선언 (var string1 = 'abc';)

  - 생성자 함수 사용 (var string2 = new String('abc');)
- 속성

속성	설명
length	문자열의 길이
prototype	모든 String 객체에 포함할 속성이나 메서드 추가 허용

2. 기본 내장 객체의 종류

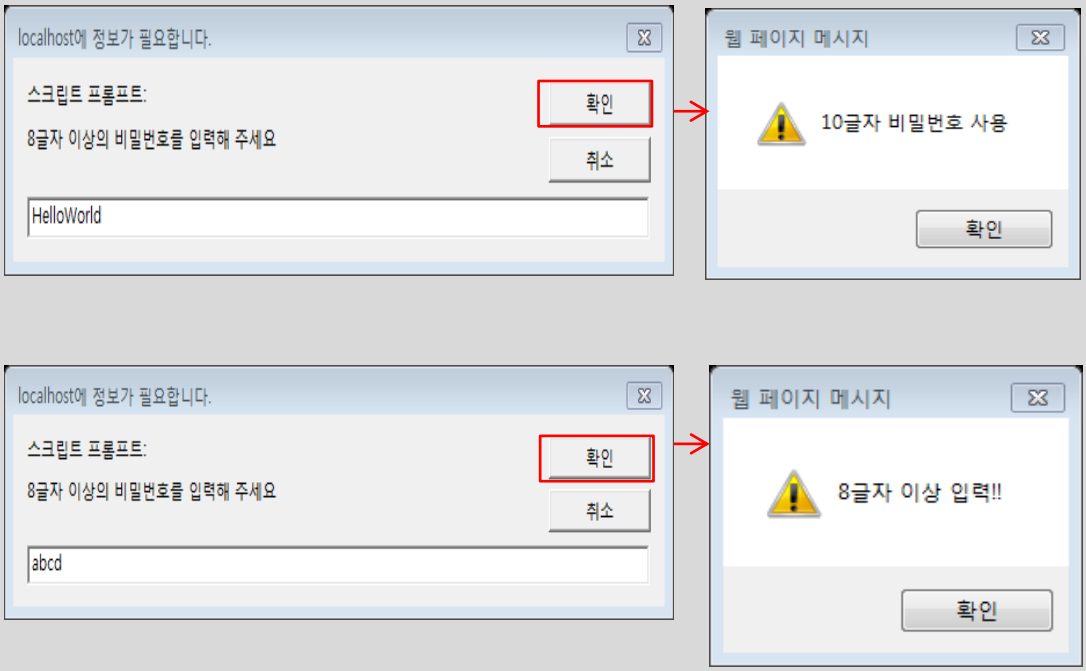
3) String 객체

- String 객체의 특징

• 예

```
<script>
    var password = prompt('8글자 이상의 비밀번호를 입력해 주세요', '');

    if (password.length < 8){
        alert('8글자 이상 입력!!');
    } else {
        alert(password.length + '글자 비밀번호 사용');
    }
</script>
```

• 출력 화면



2. 기본 내장 객체의 종류

3) String 객체

- String 객체의 특징

• 기본 메서드

메서드	설명
charAt(position)	지정된 위치의 문자를 리턴
charCodeAt(position)	지정된 위치 문자의 유니코드를 리턴
concat ()	두 개 혹은 그 이상의 매개 변수로 입력한 문자열을 이어서 리턴
indexOf()	문자열 내에서 일치하는 값의 위치를 앞에서부터 찾아 위치 리턴
lastIndexOf()	문자열 내에서 일치하는 값의 위치를 뒤에서부터 찾아 위치 리턴 (검색은 뒤에서부터 하지만 위치는 문자열 시작을 기준으로 리턴)
localeCompare()	정렬 순서에 근거하여 두 개의 문자열을 비교 str1.localeCompre(str2)의 형식으로 str1이 str2보다 먼저 정렬되면 -1, 같으면 0, str1이 str2보다 뒤에 정렬되면 1의 값을 리턴
match()	문자열 내에 매개 변수로 입력된 문자열이 있는지 확인
replace()	문자열 내에 첫 번째 매개 변수로 입력된 문자열이 있는지 확인하여 있으면, 두 번째 매개변수로 입력된 문자열로 변경하여 리턴
search()	매개 변수로 입력된 문자열과 일치하는 문자열의 위치를 리턴
slice()	특정 위치의 문자열을 추출하여 새로운 문자열 리턴
split()	매개 변수로 입력된 것을 기준으로 하여 문자열을 잘라서 리턴
substr()	첫 번째 매개 변수 위치 시작하여 두 번째 매개 변수의 수만큼 문자열을 잘라서 리턴
substring()	첫 번째 매개 변수 위치 시작하여 두 번째 매개 변수 위치까지 문자열을 잘라서 리턴
toLowerCase()	문자열을 소문자로 변환하여 리턴
toUpperCase()	문자열을 대문자로 변환하여 리턴

2. 기본 내장 객체의 종류

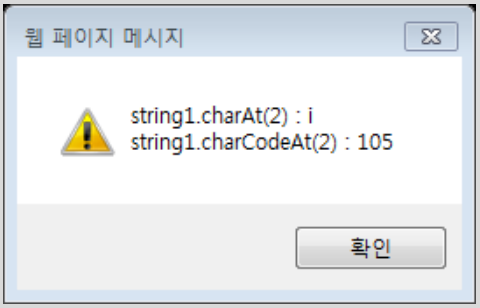
3) String 객체

- String 객체의 특징

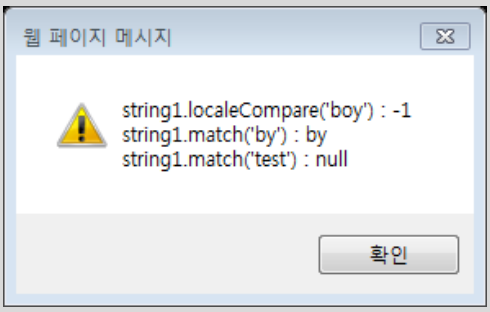
• 예

```
<script>
    var string1 = 'Ability is decided by one`s own effort.';
    var out='';
①   out+='string1.charAt(2) : '+ string1.charAt(2)+'₩n';
    out+='string1.charCodeAt(2) : '+ string1.charCodeAt(2)+'₩n';
    alert(out);

②   out="string1.concat('  test') : "+string1.concat('  test')+'₩n';
    out += "string1.indexOf('is') : " + string1.indexOf('is') + '₩n';
    out += "string1.lastIndexOf('is') : " + string1.lastIndexOf('is') + '₩n';
    alert(out);

③   out = "string1.localeCompare('boy') : " + string1.localeCompare('boy') + '₩n';
    out += "string1.match('by') : " + string1.match('by') + '₩n';
    out += "string1.match('test') : " + string1.match('test') + '₩n';
    alert(out);
```

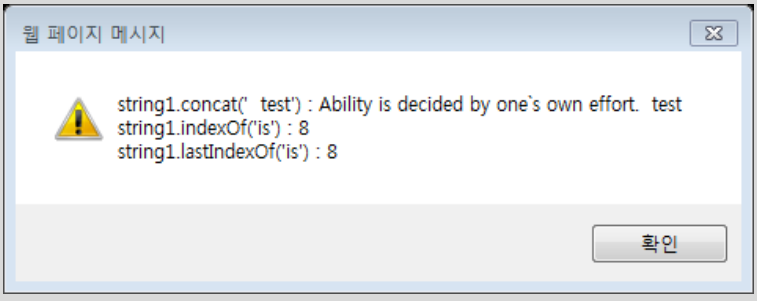
① 출력 화면



② 출력 화면



③ 출력 화면



2. 기본 내장 객체의 종류

3) String 객체

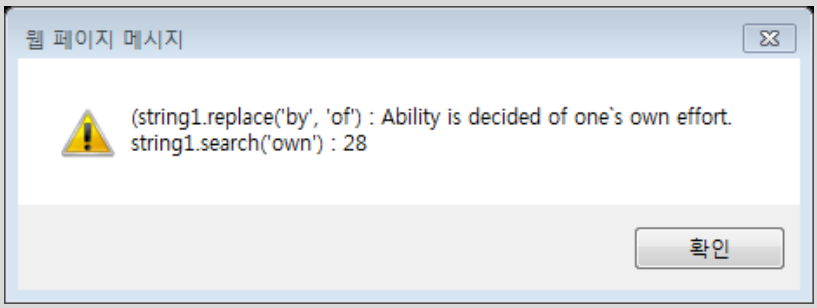
- String 객체의 특징

• 예

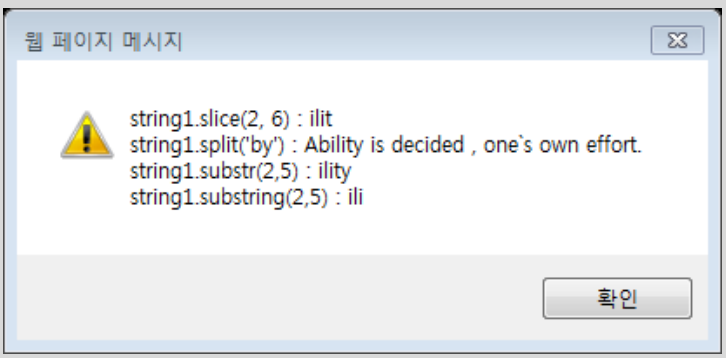
④
```
out = "(string1.replace('by', 'of') : " + string1.replace('by', 'of') + '₩n';
out += "string1.search('own') : " + string1.search('own') + '₩n';
alert(out);
```

⑤
```
out = "string1.slice(2, 6) : " + string1.slice(2, 6) + '₩n';
out += "string1.split('by') : " + string1.split('by') + '₩n';
out += "string1.substr(2,5) : " + string1.substr(2, 5) + '₩n';
out += "string1.substring(2,5) : " + string1.substring(2, 5) + '₩n';
alert(out);
```

④ 출력 화면



⑤ 출력 화면



2. 기본 내장 객체의 종류

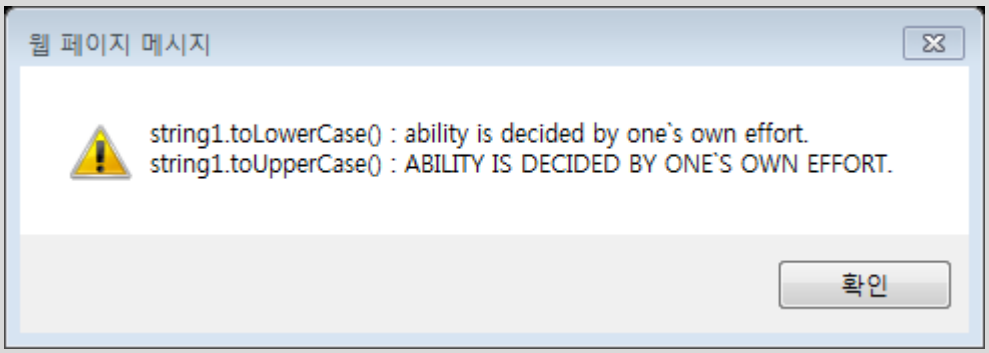
3) String 객체

- String 객체의 특징

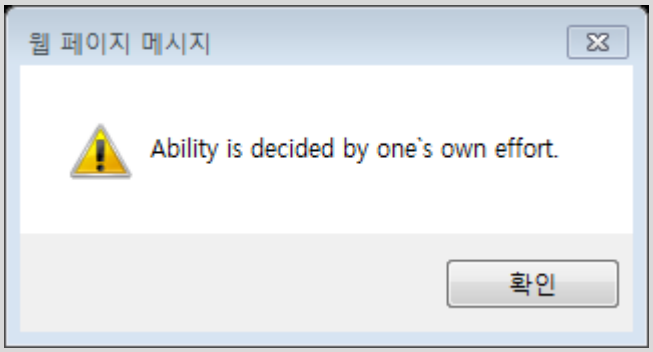
• 예

```
⑥ out = "string1.toLowerCase() : " + string1.toLowerCase() + '₩n';
   out += "string1.toUpperCase() : " + string1.toUpperCase();
   alert(out);

⑦ alert(string1);
```

⑥ 출력 화면



⑦ 출력 화면



2. 기본 내장 객체의 종류

3) String 객체

- String 객체의 특징

- HTML 관련 메서드

  - 문자열을 HTML 태그로 감싸 리턴

  - 표준 메서드가 아니기 때문에, 브라우저에 따라 원했던 결과를 못할 수도 있음

메서드	설명
anchor()	a태그로 문자열을 감싼 후 리턴
big()	big 태그로 문자열을 감싼 후 리턴
blink()	blank 태그로 문자열을 감싼 후 리턴
bold()	b 태그로 문자열을 감싼 후 리턴
fixed()	tt 태그로 문자열을 감싼 후 리턴
fontcolor()	font 태그로 문자열을 감싼 후 color 속성을 주어 리턴
fontsize()	font 태그로 문자열을 감싼 후 size 속성을 주어 리턴
italics()	I 태그로 문자열을 감싼 후 리턴
link()	a 태그로 문자열을 감싼 후 href 속성을 지정해 리턴
small()	small 태그로 문자열을 감싼 후 리턴
strike()	strike 태그로 문자열을 감싼 후 리턴
sub()	sub 태그로 문자열을 감싼 후 리턴
sup()	sup 태그로 문자열을 감싼 후 리턴

▣ 정리하기

1. 내장 객체의 개요

- 내장 객체 : 내장 함수와 같이 JavaScript 자체에서 제공하는 객체
- 기본 자료형에서 속성과 메서드를 사용하면 일시적으로 객체로 변환

2. 기본 내장 객체의 종류

- Object 객체 : JavaScript의 최상위 객체, 모든 객체는 Object 객체를 기본으로 하여 생성
- Number 객체 : 숫자를 표현할 때 사용하는 객체, 숫자에 관련한 속성과 메서드 제공
- String 객체 : 문자열을 표현할 때 사용하는 객체, length 속성과 다양한 메서드 제공
- 메서드 체이닝 : 메서드를 연속해서 사용하는 것