Chapter13
JavaScript 회원정보
입력양식 만들기

HTML & JavaScript

Contents

Form 객체

❖ Form 객체

- 입력 양식의 틀이 되는 <FORM>태그에 접근할 수 있도록 지원
- Document 객체의 하위에 위치
- 속성들은 모두 <FORM> 태그의 속성들의 정보에 관련된 것
- <FORM> 태그 안의 여러 입력 양식을 요소라고 함

```
document.폼 이름.elements[인덱스 번호]
```

Form 객체의 속성

속성	설명
action	〈FORM〉 태그의 ACTION 속성 정보
method	〈FORM〉 태그의 METHOD 속성 정보
elements	〈FORM〉 태그의 양식 배열 저장
encoding	〈FORM〉 태그의 ENCTYPE 속성 정보
length	〈FORM〉 태그의 입력 양식의 개수
name	〈FORM〉 태그의 NAME 속성 정보
target	〈FORM〉 태그의 TARGET 속성 정보
form, forms	〈FORM〉 태그의 정보

Form 객체의 메소드, 이벤트핸들러

메소드	설명
blur()	커서 제거
submit()	양식의 내용 전송
reset()	양식에 입력된 내용 초기화

이벤트 핸들러	설명
onSubmit	Submit 버튼을 클릭할 경우 발생
onReset	Reset 버튼을 클릭할 경우 발생

Form 객체

형식 :

```
<FORM NAME = "이름" TARGET = "창 이름"
   ACTION = "URL" ENCTYPE = "형식"
   METHOD = GET | POST
   onSubmit = "이벤트" onReset = "이벤트">
.....
</FORM>
document.폼 이름.[속성 | 메소드]
```

속성 : **action, method, elements, encoding, length, name, target, form, forms**

메소드 : **blur(), submit(), reset()**

이벤트 핸들러 : **onSubmit, OnReset**

Form 객체

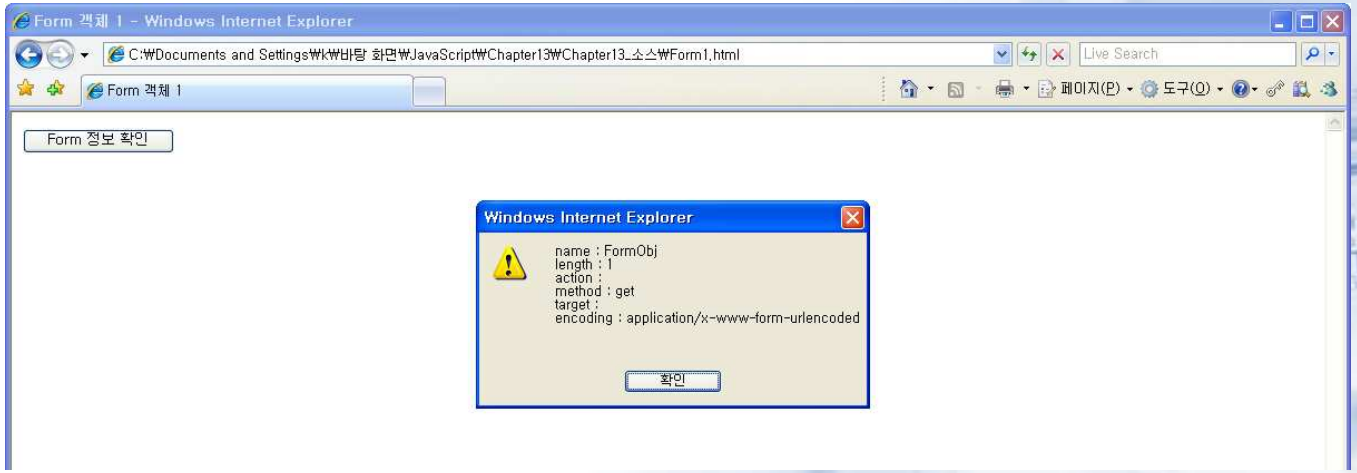
예제 :

```
<SCRIPT LANGUAGE="JavaScript">
<!--
 function Output()
 {
  var strInfo = "name : " + document.FormObj.name + "\n";
  strInfo += "length : " + document.FormObj.length + "\n";
  strInfo += "action : " + document.FormObj.action + "\n";
  document.FormObj.submit();
  alert(strInfo)
 }
//-->
</SCRIPT>
<FORM NAME="FormObj">
 <INPUT TYPE="button" VALUE="Form 정보  확인" onClick="Output()">
</FORM>
```

프로그램 13-1

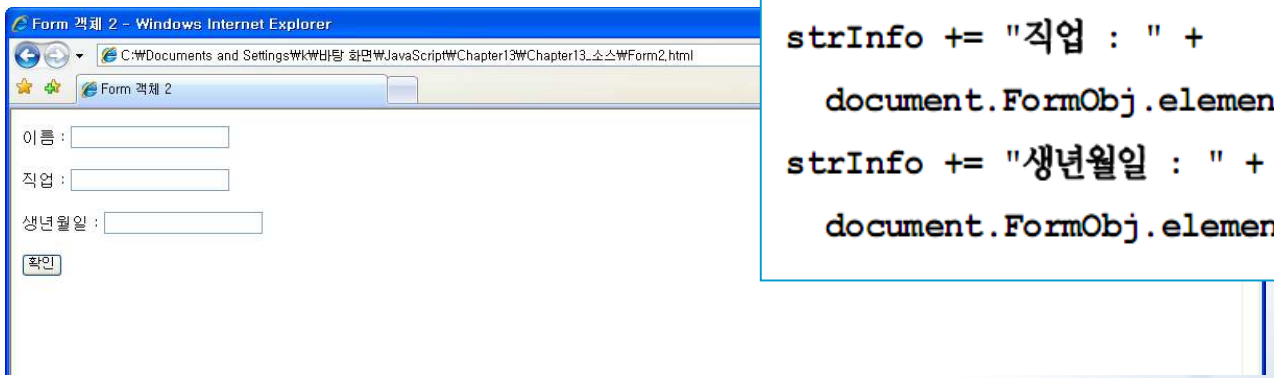
```
var strInfo = "name : "
    + document.FormObj.name + "\n";
strInfo += "length : "
    + document.FormObj.length + "\n";
strInfo += "action : "
    + document.FormObj.action + "\n";
```

[그림 13-1] Form1의 실행결과



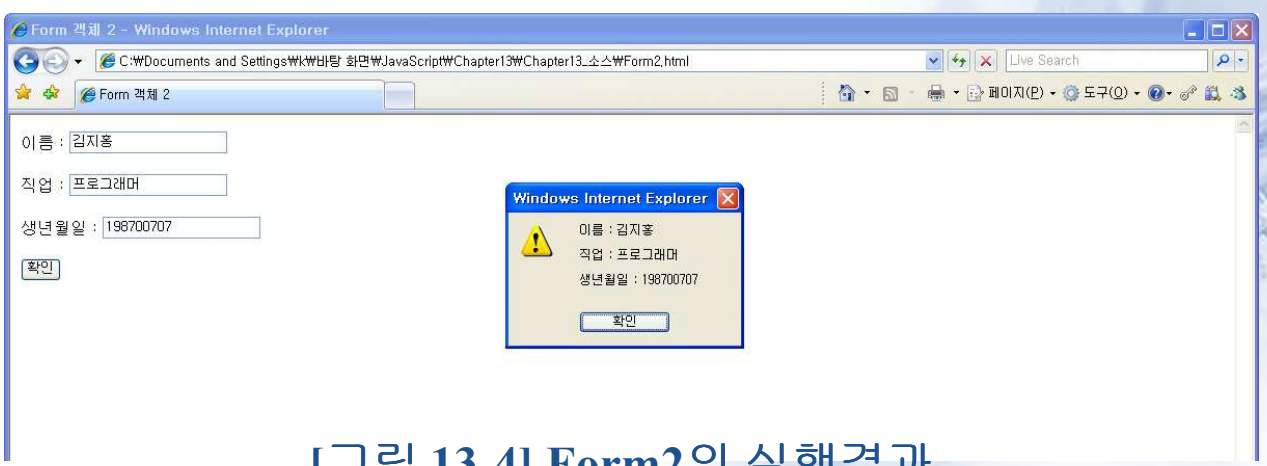
[그림 13-2] Form1의 실행결과

프로그램 13-2




```
strInfo = "이름 : " +
  document.FormObj.elements[0].value + "\n\n";
strInfo += "직업 : " +
  document.FormObj.elements[1].value + "\n\n";
strInfo += "생년월일 : " +
  document.FormObj.elements[2].value;
```

[그림 13-3] Form2의 실행결과



[그림 13-4] Form2의 실행결과

Text 객체

❖Text 객체

- <INPUT> 태그의 TYPE 속성의 값으로 text를 입력하여 만들어진 텍스트 박스에 접근
- Form 객체에서 파생

형식 :

```
<INPIUT TYPE = "text" NAME = "이름" VALUE ="값"
   SIZE = "크기" MAXLENGTH = "크기"
   onBlur = "이벤트" onChange = "이벤트"
   onFocus = "이벤트" onSelect = "이벤트">
```

속성 : **defaultValue, name, value**

메소드 : **blur(), focus(), select()**

이벤트 핸들러 : **onBlur, onChange, onFocus, onSelect**

Text 객체

예제 :

```
<SCRIPT LANGUAGE="JavaScript">
<!--
 function Output(strData)
 {
  if (strData.id.value == "이름  입력")
  {
   alert("이름을  입력하세요");
   form.id.focus();
  }
  else
  {
   var strInfo = "name : " + strData.id.name + "\n";
   strInfo += "value : " + strData.id.value + "\n";
   strInfo += "defaultValue : " + strData.id.defaultValue + "\n\n";
   alert(strInfo)
  }
 }
```

Text 객체의 속성, 메소드, 이벤트 핸들러

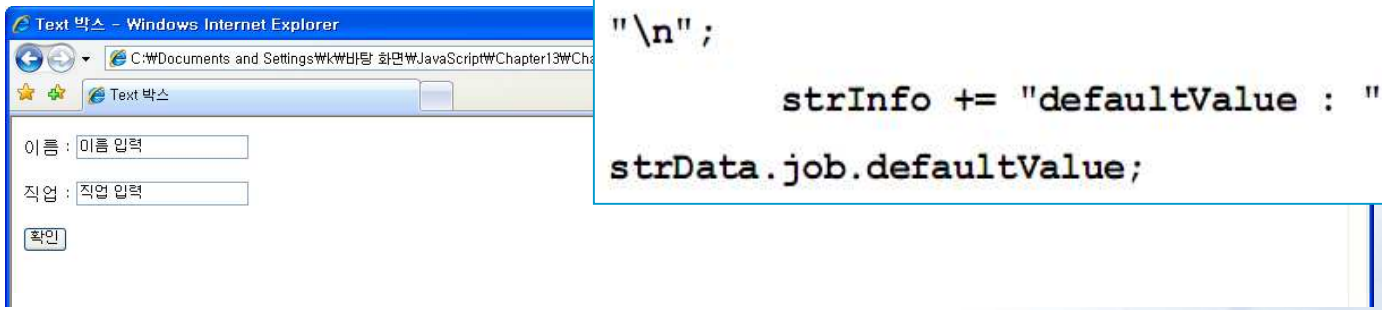
속성	설명
defaultValue	초기 텍스트 박스에 설정된 문자열 지정
name	텍스트 박스의 이름
value	텍스트 박스의 문자열 값

메소드	설명
blur()	커서 제거
focus()	텍스트 박스에 커서 지정
select()	텍스트 박스의 문자열 선택

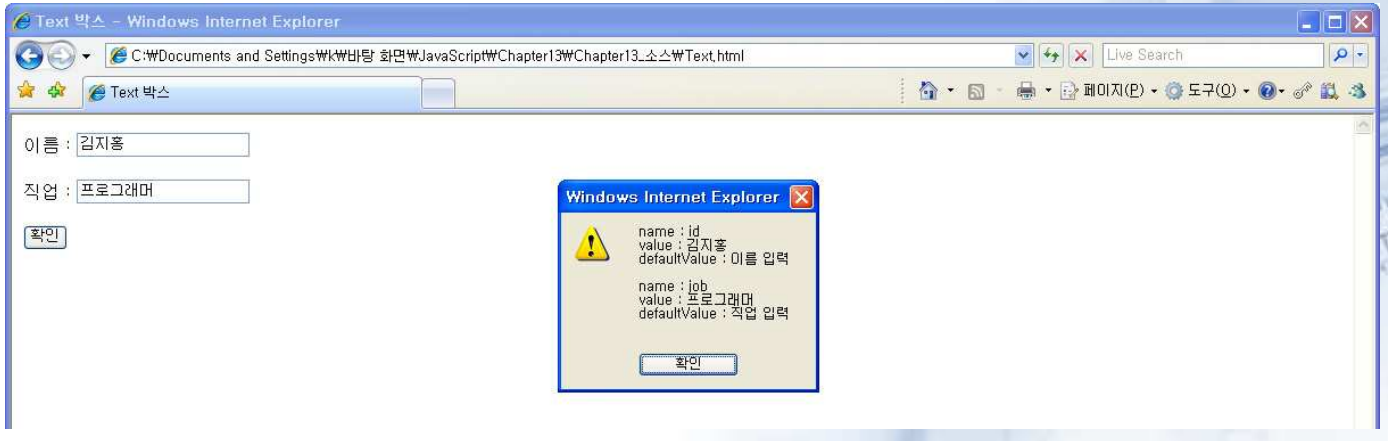
이벤트 핸들러	설명
onBlur	현재 텍스트 박스의 포커스를 벗어날 때 발생
onChange	현재 텍스트 박스 안에 있는 내용이 바뀔 때 발생
onFocus	텍스트 박스에 포커스가 설정되었을 때 발생
onSelect	텍스트 박스를 클릭하거나 선택하였을 때 발생

프로그램 13-3

```
        strInfo += "name : " + strData.job.name +
"\n";
        strInfo += "value : " + strData.job.value +
"\n";
        strInfo += "defaultValue : " +
strData.job.defaultValue;
```



[그림 13-5] Text의 실행결과



[그림 13-6] Text의 실행결과

Password 객체

❖ Password 객체

- <INPUT> 태그의 TYPE 값을 password로 지정한 텍스트 박스에 접근할 경우 사용
- 속성, 메소드들은 기본적으로 Text 객체의 것들과 같음
- Password 텍스트 박스에 어떠한 정보를 입력하였을 경우 value 속성을 사용하여 해당 값을 가져올 수 없음

형식 :

```
<INPUT TYPE = "password" NAME = "이름"
   VALUE = "값" SIZE = "크기">
```

속성 : defaultValue, name, value

메소드 : blur(), focus(), select()

Password 객체

예제 :

```
function Output(strData)
 {
  if (strData.id.value == "")
  {
   alert("비밀번호를  입력하세요");
   form.pwd.focus();
  }
  else
  {
   var strInfo = "name : " + strData.pwd.name + "\n";
   strInfo += "value : " + strData.pwd.value + "\n";
   strInfo += "defaultValue : " + strData.pwd.defaultValue +
"\n\n";
   alert(strInfo)
  }
 }
```

Password 객체의 속성, 메소드, 이벤트 핸들러

속성	설명
defaultValue	초기 Password 텍스트 박스에 설정된 문자열 지정
name	Password 텍스트 박스의 이름
value	Password 텍스트 박스의 문자열 값

메소드	설명
blur()	커서 제거
focus()	Password 텍스트 박스에 커서 지정
select()	Password 텍스트 박스의 문자열 선택

TextArea 객체

❖ TextArea 객체

- TYPE 속성의 값으로 textarea를 입력하면 여러 줄을 입력할 수 있는 텍스트 박스에 접근할 때 사용

형식 :

```
<TEXTAREA NAME = "이름" ROWS = "크기" COLS = "크기"
   onBlur = "이벤트" onChange = "이벤트"
   onFocus = "이벤트" onSelect = "이벤트"
```

속성 : defaultValue, name, value

메소드 : blur(), focus(), select()

이벤트 핸들러 : onBlur, onChange, onFocus, onSelect

TextArea 객체

예제 :

```
function Output(form) {
  var strInfo = "name : " + form.Introduce.n
  strInfo += "value : \n" + form.Introduce.v
  alert(strInfo);
}
function Output_Status(strInfo)
{
  window.status = strInfo;
}
```

TextArea 객체의 속성, 메소드, 이벤트 핸들러

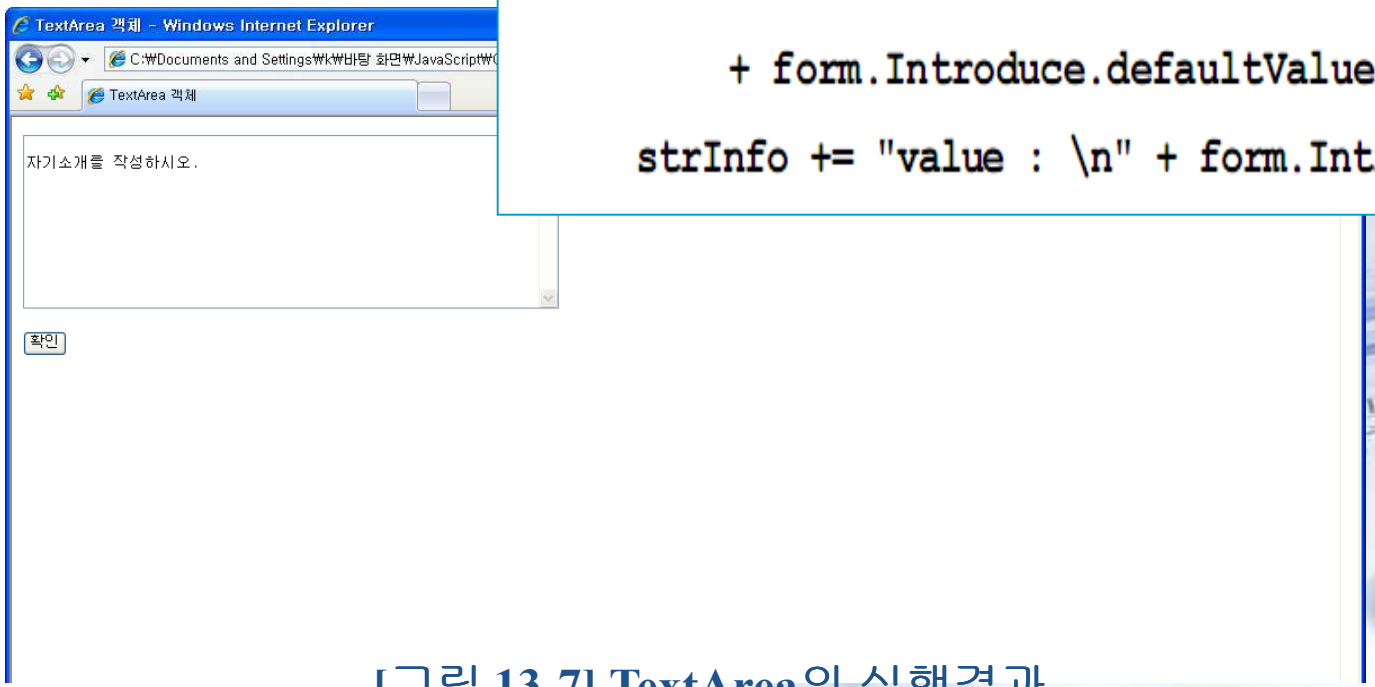
속성	설명
defaultValue	초기 TEXTAREA 입력 양식에 설정된 문자열 지정
name	TEXTAREA 객체의 이름
value	TEXTAREA 입력 양식의 문자열 값

메소드	설명
blur()	커서 제거
focus()	특정 TEXTAREA 입력 양식에 커서 지정
select()	TEXTAREA 입력 양식의 문자열 선택

이벤트 핸들러	설명
onBlur	현재 입력 양식의 포커스를 벗어날 때 발생
onChange	현재 입력 양식 안에 있는 내용이 바뀔 때 발생
onFocus	TEXTAREA 입력 양식에 포커스가 설정되었을 때 발생
onSelect	TEXTAREA 입력부분을 클릭하거나 선택하였을 때 발생

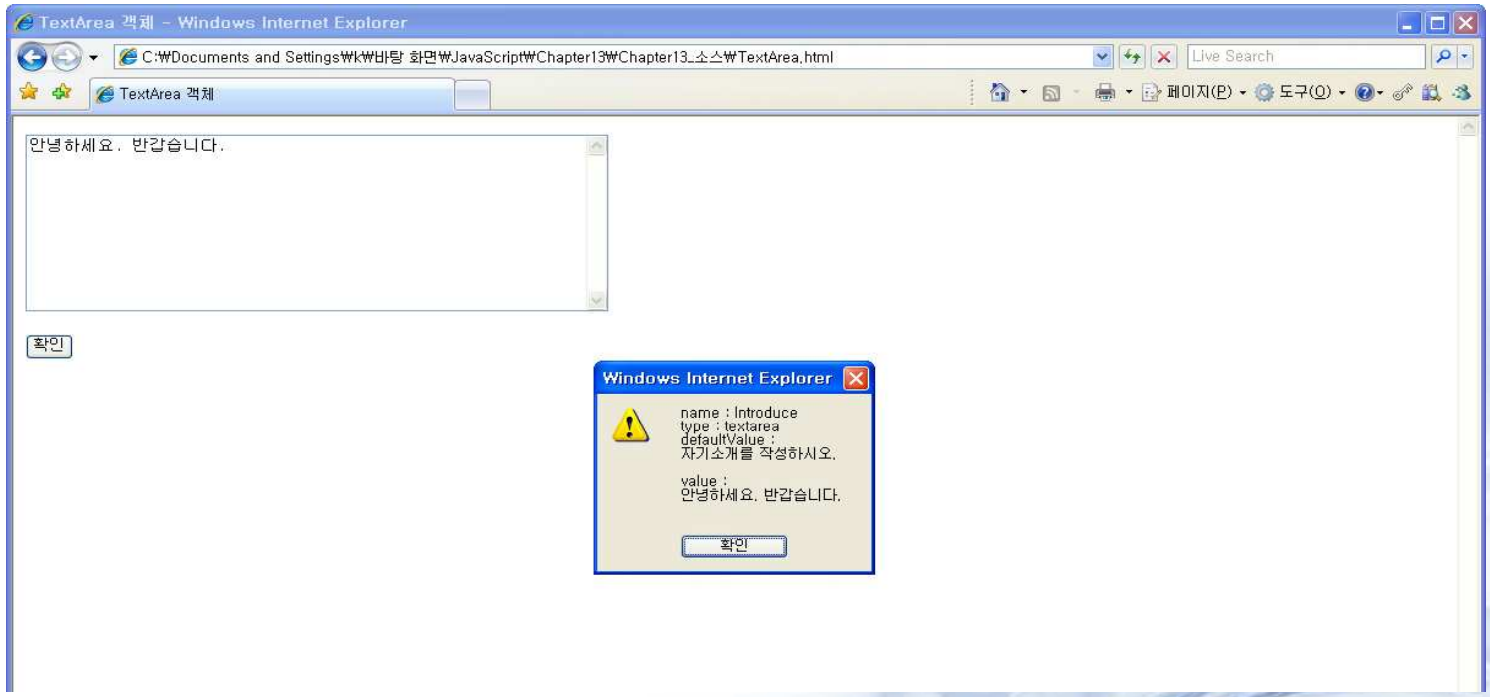
프로그램 13-4

```
        var strInfo = "name : " + form.Introduce.name
+ "\n";
        strInfo += "type : " + form.Introduce.type +
"\n";
        strInfo += "defaultValue :"
            + form.Introduce.defaultValue + "\n";
        strInfo += "value : \n" + form.Introduce.value;
```



[그림 13-7] TextArea의 실행결과

프로그램 13-4



[그림 13-8] TextArea의 실행결과

Hidden 객체

❖ Hidden 객체

- <INPUT> 태그의 TYPE 속성의 값으로 hidden으로 지정하여 만든 입력양식에 접근할 때 사용

형식 :

```
<INPUT TYPE = "hidden" NAME = "이름" VALUE = "값">
```

속성 : defaultValue, name, value

예제 :

```
 function Output(strData)
  {
    var strInfo = "name : " + strData.hiddenInfo.name + "\n";
    strInfo += "value : " + strData.hiddenInfo.value + "\n";
    strInfo += "defaultValue : " + strData.hiddenInfo.defaultValue +
"\n\n";
     alert(strInfo)
  }
```

Hidden 객체의 속성

속성	설명
defaultValue	초기 Hidden 입력 양식에 설정된 문자열 지정
name	Hidden 입력 양식의 이름
value	Hidden 입력 양식의 문자열 값

❖ Button 객체

- <INPUT> 태그의 TYPE 속성의 값으로 button을 입력하여 만든 버튼에 접근할 때 사용
- 속성으로는 버튼의 이름을 나타내는 name 속성, 버튼의 값을 나타내는 value 속성이 있음
- 버튼을 클릭한 것과 같은 효과를 내는 click() 메소드가 있으며 버튼을 클릭하였을 경우 발생하는 이벤트를 처리하기 위한 onClick 이벤트 핸들러가 있음

Button 객체

형식 :
<INPUT TYPE = "button" NAME = "이름" VALUE = "값" onClick = "이벤트">

속성 : name, value

메소드 : click()

이벤트 핸들러 : onClick

예제 :

```
function Output(strData)
 {
   var strInfo = "이름  : " + strData.name + "\n";
   strInfo += "값  : " + strData.value;
   alert(strInfo);
 }
```

Button 객체의 속성, 메소드, 이벤트 핸들러

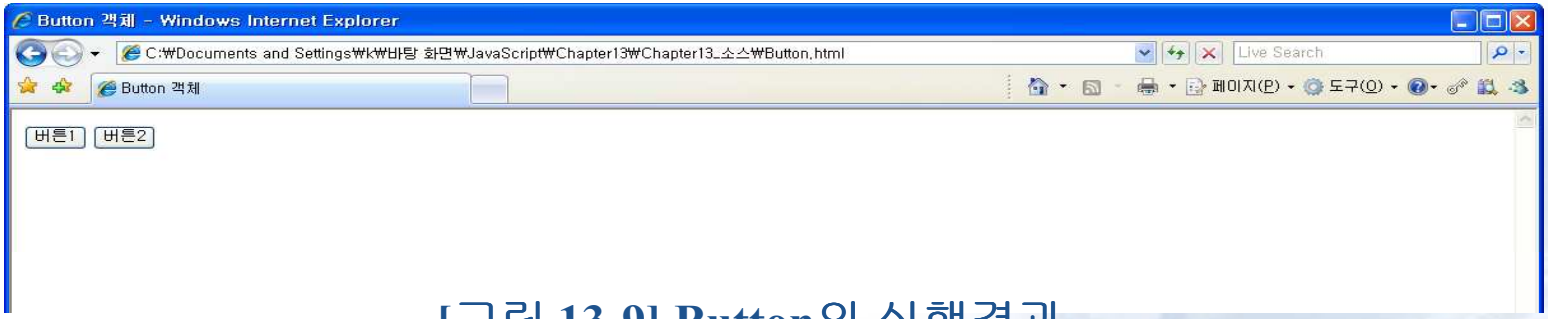
속성	설명
name	버튼의 이름
value	버튼의 값

메소드	설명
click()	버튼 클릭

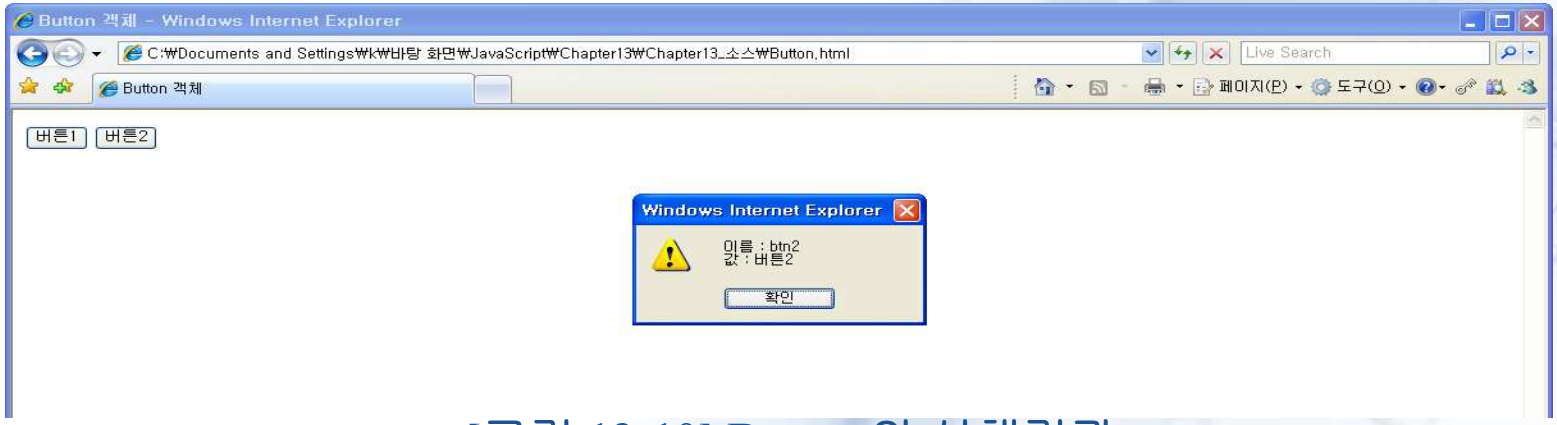
이벤트 핸들러	설명
onClick	버튼을 클릭하였을 때 발생

프로그램 13-5

```
var strInfo = "이름 : " + strData.name + "\n";
strInfo += "값 : " + strData.value;
```



[그림 13-9] Button의 실행결과



[그림 13-10] Button의 실행결과

Checkbox 객체

❖ Checkbox 객체

- <INPUT> 태그의 TYPE 속성의 값으로 checkbox를 입력하여 만든 체크박스에 접근하고자 할 때 사용하는 객체

Checkbox 객체

형식 :

```
<INPUT TYPE = "checkbox" NAME = "이름" VALUE = "값"
   CHECKED onClick = "이벤트">
```

속성 : checked, defaultChecked, name, value

메소드 : click()

이벤트 핸들러 : onClick

예제 :

```
 function Output(form)
  {
    var strInfo = "당신의  취미는  \n";
    if(form.check1.checked)
    strInfo += "게임  \n";
    if(form.check2.checked)
    strInfo += "독서  \n";
    strInfo += "입니다. \n";
}
```

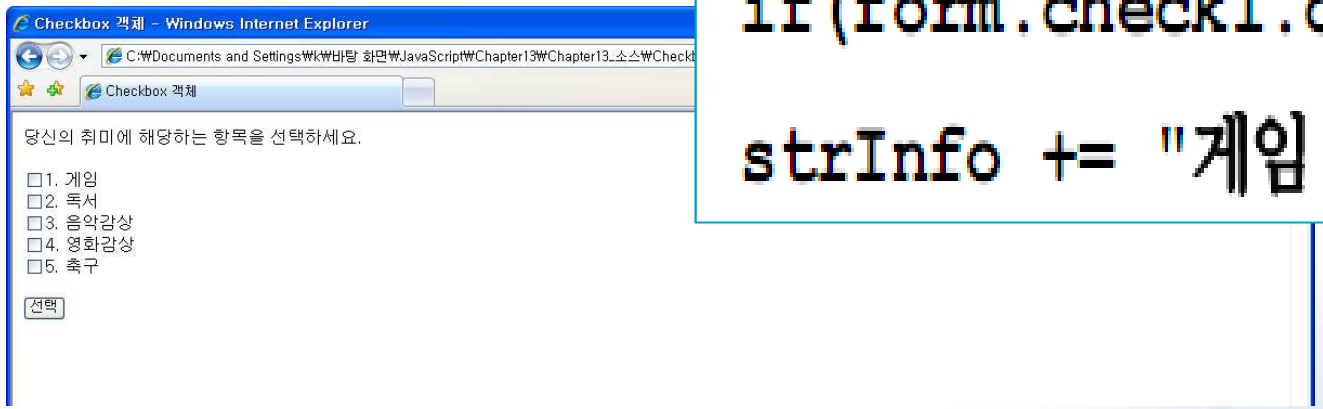
Checkbox 객체의 속성, 메소드, 이벤트 핸들러

속성	설명
checked	체크박스가 선택되었는지의 여부
defaultChecked	체크박스가 처음 선택되었는지의 여부
name	체크박스의 이름
value	체크박스 입력 양식의 문자열 값

메소드	설명
click()	체크박스 클릭

이벤트 핸들러	설명
onClick	체크박스를 클릭하였을 때 발생

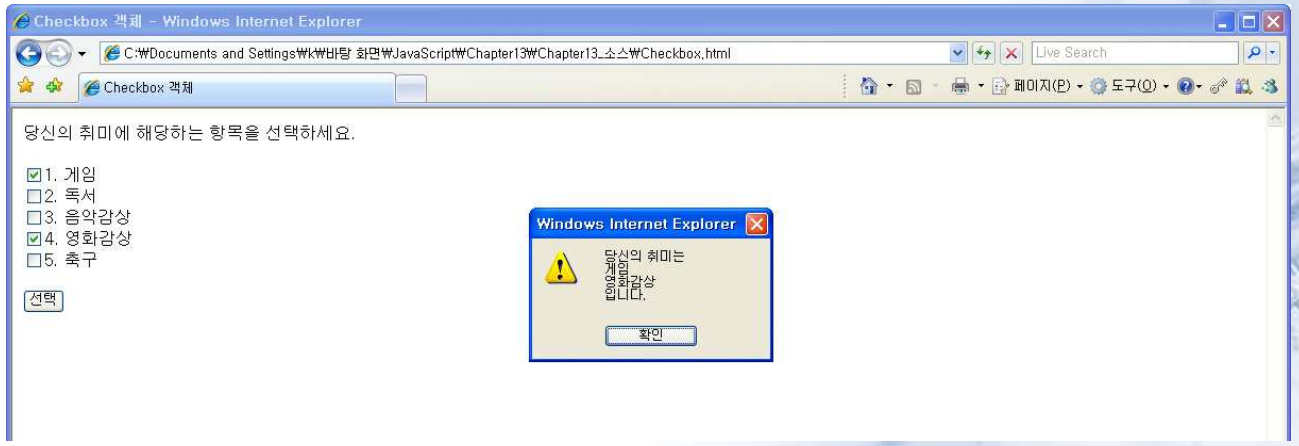
프로그램 13-6




```
if(form.check1.checked)

strInfo += "게임 \n";
```

[그림 13-11] Checkbox의 실행결과



[그림 13-12] Checkbox의 실행결과

❖ Radio 객체

- <INPUT> 태그의 TYPE 속성의 값으로 Radio 를 입력하여 만든 체크박스에 접근하고자 할 때 사용하는 객체
- 속성, 메소드, 이벤트 핸들러들은 모두 Checkbox 객체의 속성, 메소드, 이벤트 핸들러들과 같음
- 여기에 한 가지 속성이 더 추가되어 있는데 바로 length 속성이 있음

Radio 객체

형식 :

```
<INPUT TYPE = "radio" NAME = "이름" VALUE = "값"
   CHECKED onClick = "이벤트">
```

속성 : checked, defaultChecked, length, name, value

메소드 : click()

이벤트 핸들러 : onClick

예제 :

```
function Output(form)
 {
   for(var i = 0; i < form.hobby.length; i++) {
     if(form.hobby[i].checked == true) {
         alert("당신의 취미는 " + form.hobby[i].value + " 입니다.");
     }
   }
 }
```

Radio 객체의 속성, 메소드, 이벤트 핸들러

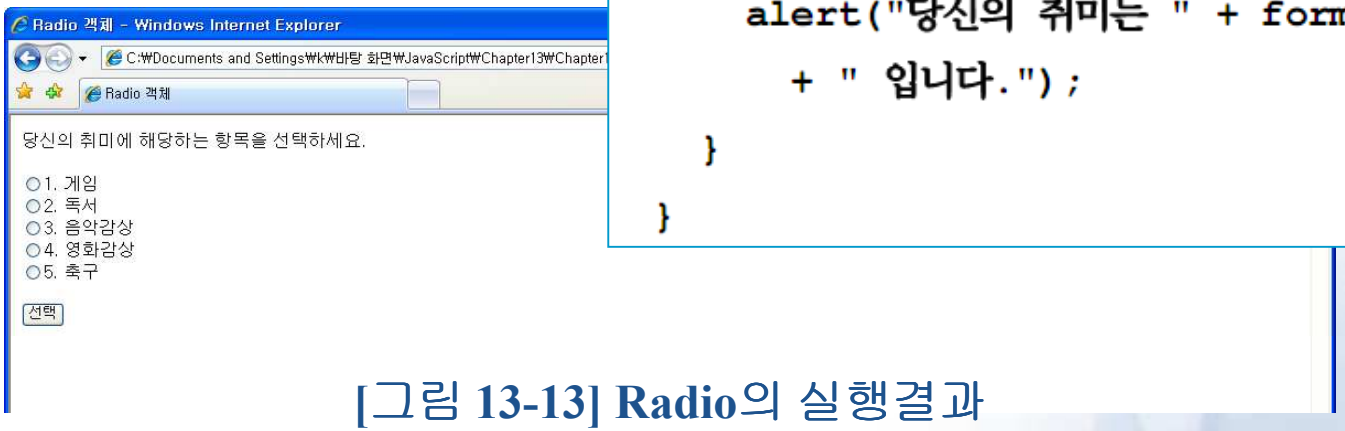
속성	설명
checked	라디오 버튼가 선택되었는지의 여부
defaultChecked	라디오 버튼가 처음 선택되었는지의 여부
length	그룹 안에 포함되어 있는 라디오 버튼 개수
name	라디오 버튼의 이름
value	라디오 버튼 입력 양식의 문자열 값

메소드	설명
click()	라디오 버튼 클릭

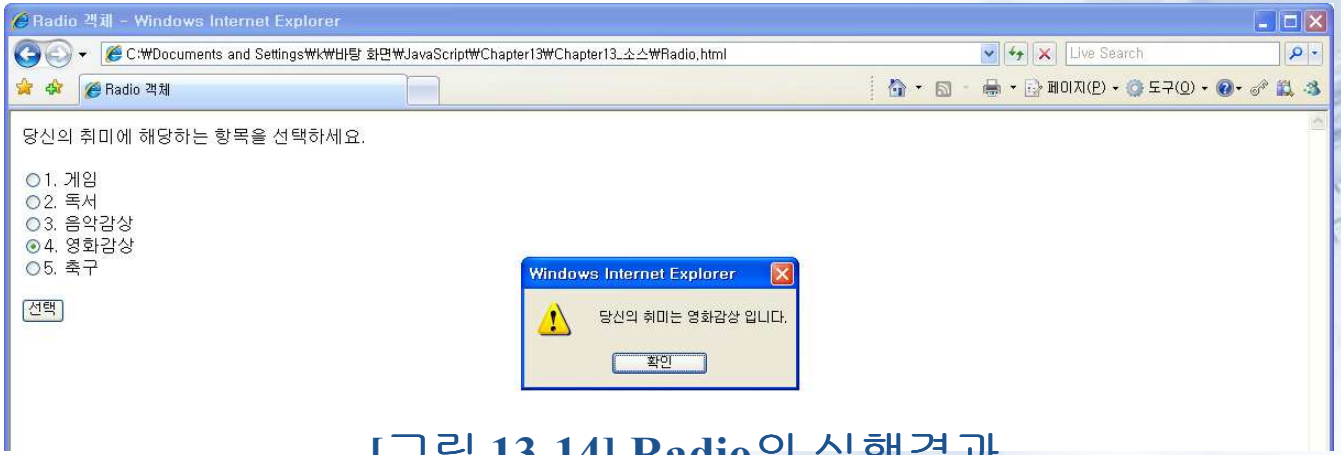
이벤트 핸들러	설명
onClick	라디오 버튼을 클릭하였을 때 발생

프로그램 13-7

```
for(var i = 0; i < form.hobby.length; i++) {
  if(form.hobby[i].checked == true) {
    alert("당신의 취미는 " + form.hobby[i].value
      + " 입니다.");
  }
}
```



[그림 13-13] Radio의 실행결과



[그림 13-14] Radio의 실행결과

Select 객체

❖Select 객체

- <SELECT> 태그와 <OPTION> 태그를 사용하여 만든 콤보박스에 접근할 때 사용
- selectIndex 속성으로 콤보 박스에서 현재 선택되어 있는 항목의 인덱스를 알 수 있음
- 콤보 박스에 MULTIPLE 속성이 있을 경우 동시에 여러 개의 항목을 선택할 수 있는데 이때 selectedIndex 속성을 사용하면 선택된 항목 중 제일 첫 번째 항목의 인덱스만을 가져오게 됨
- 콤보 박스의 각 항목의 정보를 가져오기 위해서는 options 속성을 사용

Select 객체

형식 :

```
<SELECT NAME = "이름" SIZE = "크기" MULTIPLE
      onFocus = "이벤트" onBlur = "이벤트" OnChange = "이벤트">
   <OPTION SELECTED VALUE = "값" > 항목
   <OPTION SELECTED VALUE = "값" > 항목
   ......
</SELECT>
```

속성 :

name, length, selectedIndex, options, options[i].index, options[i].text, options[i].value, options[i].selected, options[i].defaultSelected

메소드 : **onChange**

Select 객체

예제 :

```
function Output(strData)
 {
  var strInfo = "";
  strInfo += "name : "+ strData.name+"\n";
  strInfo += "length : "+ strData.length + "\n";
  for(var i=0; i < strData.options.length; i++)
  {
      if(strData.options[i].selected)
        strInfo += strData.options[i].text + "\n";
  }
  alert(strInfo);
 }
```

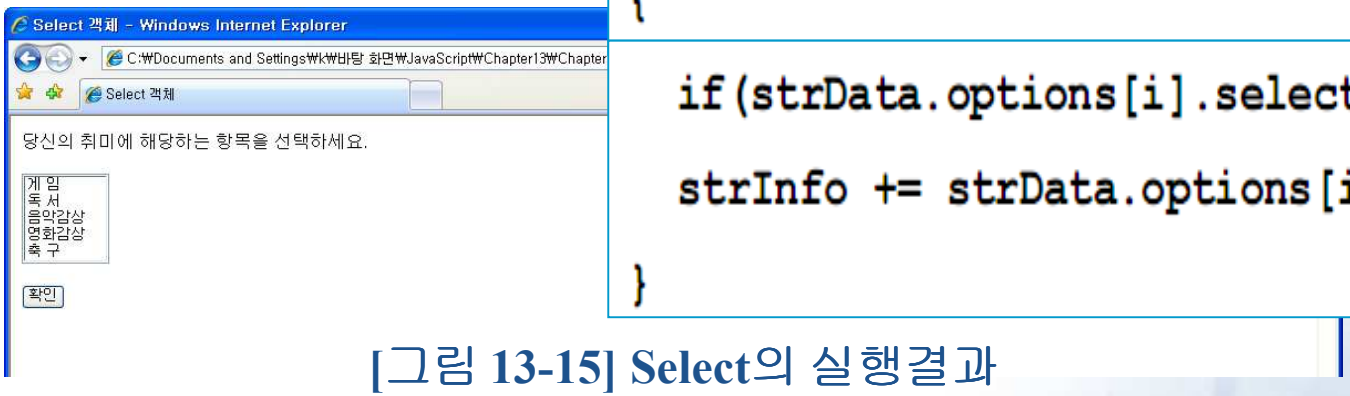
Select 객체의 속성, 이벤트 핸들러

속성	설명
name	리스트 박스의 이름
length	리스트 박스 안에 포함되어 있는 항목의 개수
selectedIndex	리스트 박스에 선택된 항목의 인덱스
options	리스트 박스에 포함된 항목 정보
options[i].index	항목의 인덱스
options[i].text	항목의 문자열
options[i].value	항목의 값
options[i].selected	항목의 선택 여부
options[i].defaultSelected	항목의 초기 선택 여부

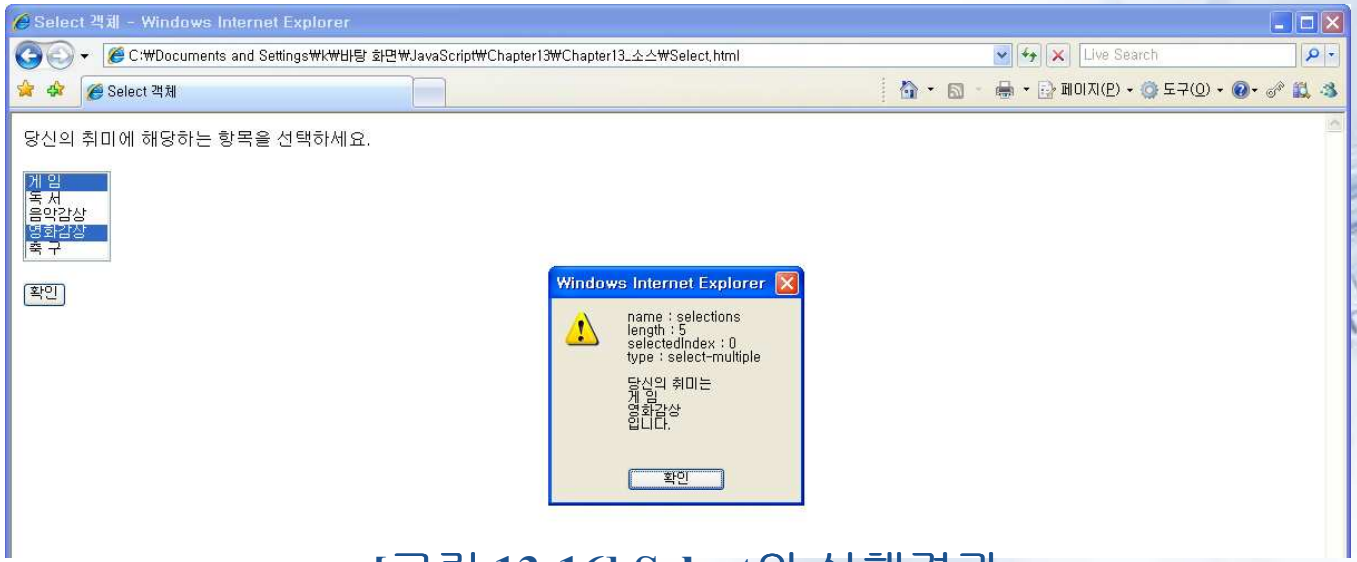
이벤트 핸들러	설명
onChange	다른 항목을 선택하였을 경우 발생하는 이벤트

프로그램 13-8

```
for(var i=0; i < strData.options.length; i++)
{
  if(strData.options[i].selected)
  strInfo += strData.options[i].text + "\n";
}
```



[그림 13-15] Select의 실행결과



[그림 13-16] Select의 실행결과

FileUpload 객체

❖ FileUpload 객체

- <INPUT> 태그의 TYPE 속성의 값으로 file을 입력하여 만든 파일 업로드 버튼에 접근할 때 사용

형식 : <INPUT TYPE = "file" NAME = ""이름>

속성 : name, value

예제 :

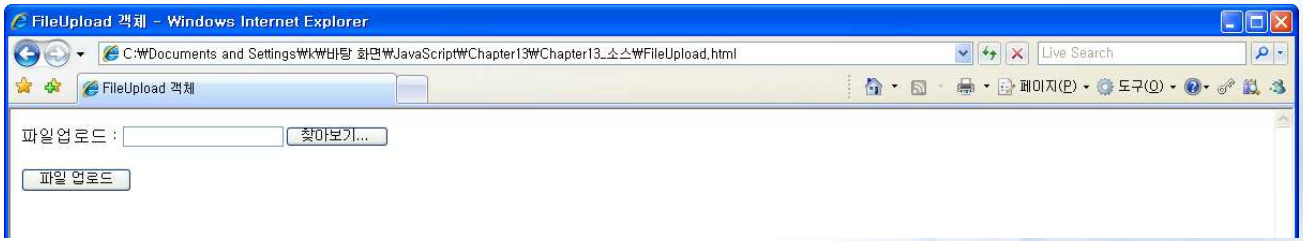
```
function Output(form)
    strInfo += form.fileup.value + "\n";
    alert(strInfo);
 }
```

FileUpload 객체의 속성

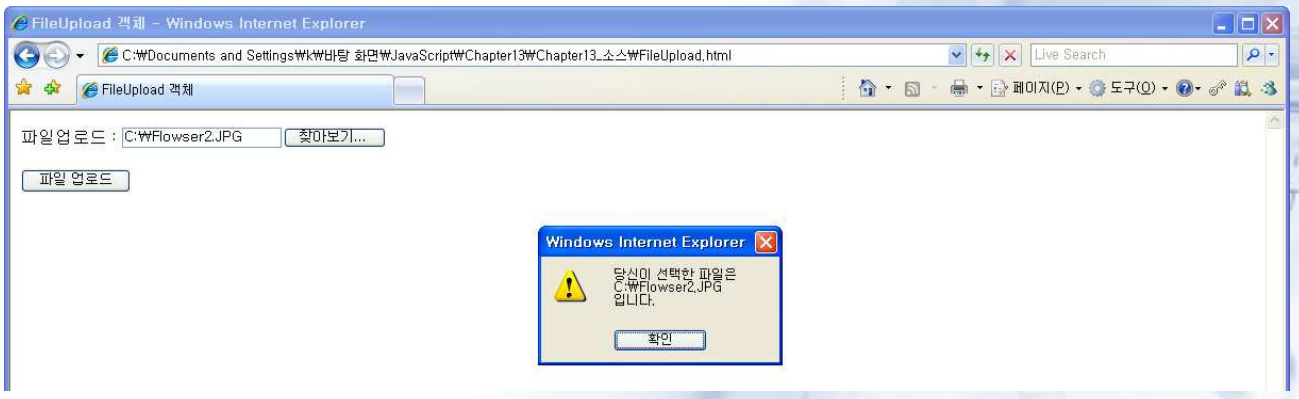
속성	설명
name	파일 업로드 입력 양식의 이름
value	파일 업로드 입력 양식에 입력된 파일 이름

프로그램 13-9

```
strInfo += form.fileup.value + "\n";
```

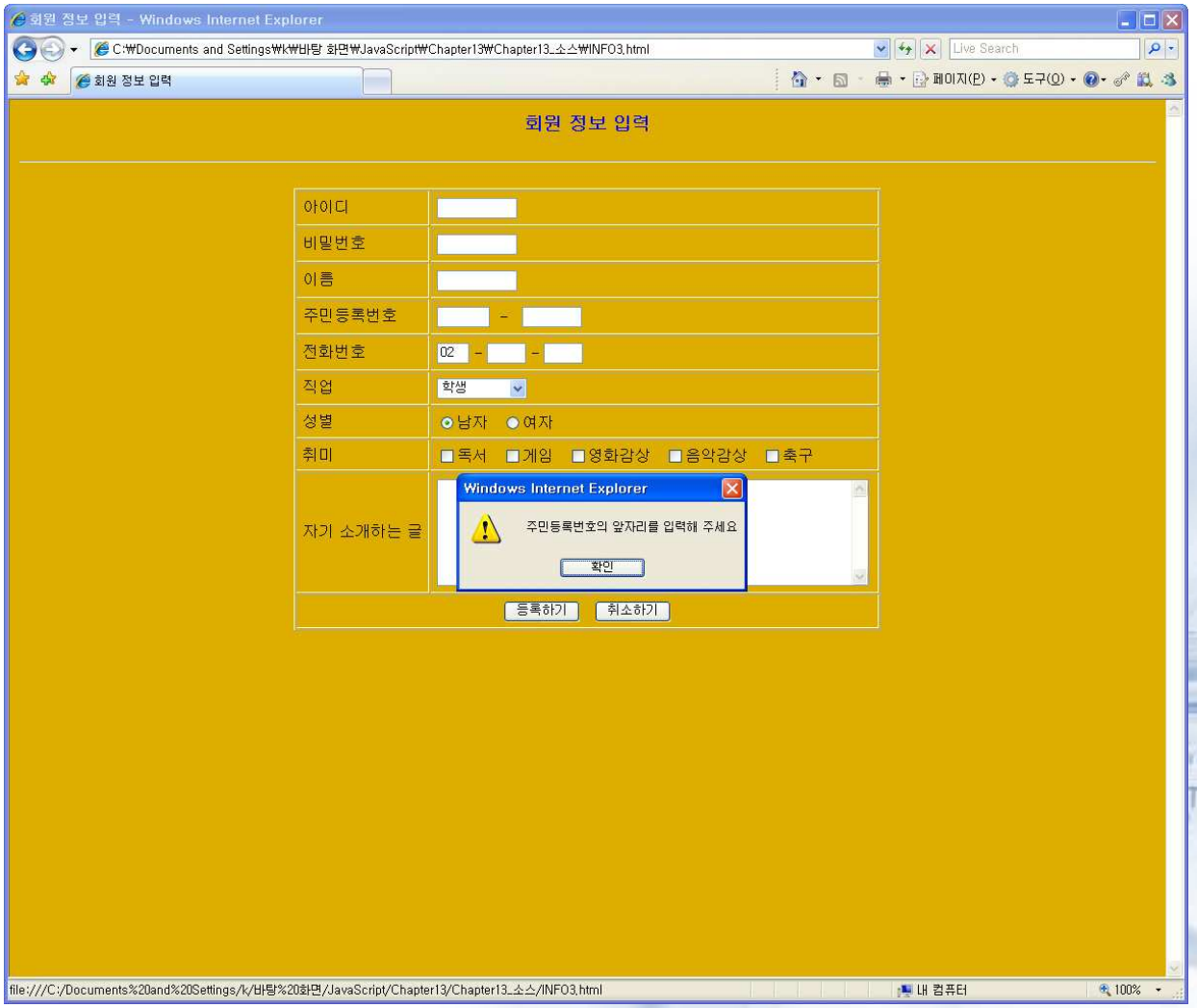


[그림 13-17] FileUpload의 실행결과



[그림 13-18] FileUpload의 실행결과

프로그램 13-10



[그림 13-19] INFO3의 실행결과