



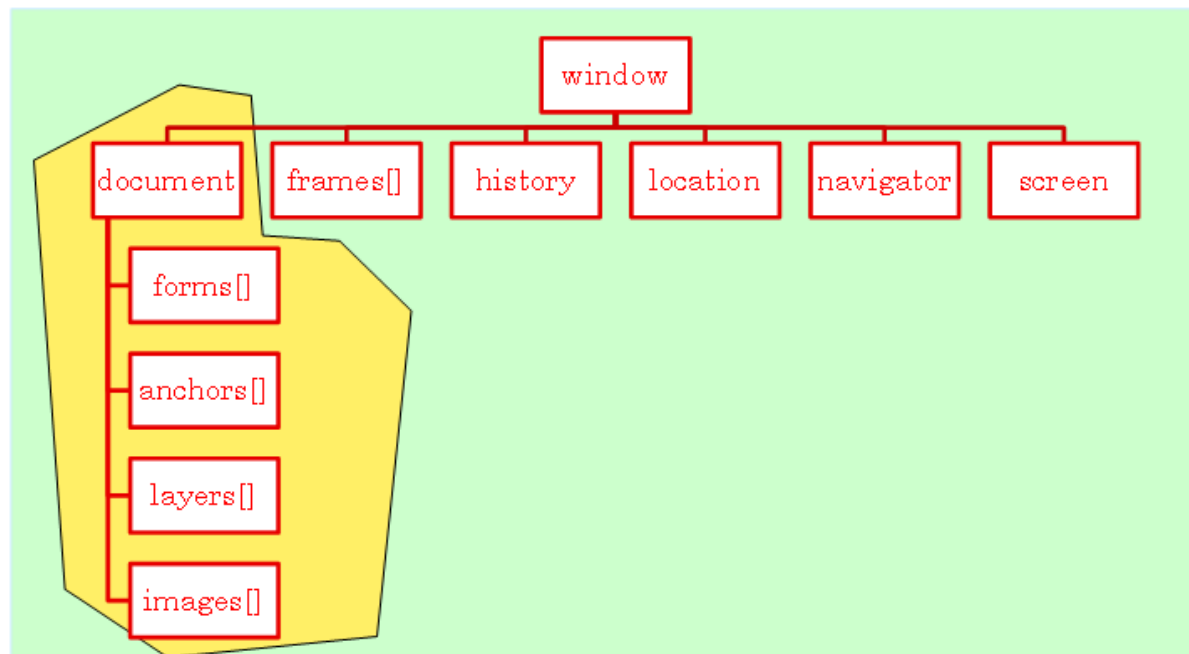
웹 프로그래밍

10. DOM과 이벤트 처리

DOM과 BOM

- BOM(Browser Object Model)
 - 웹 브라우저를 객체로 표현한 것

BOM(Browser Object Model)



DOM(Document Object Model)

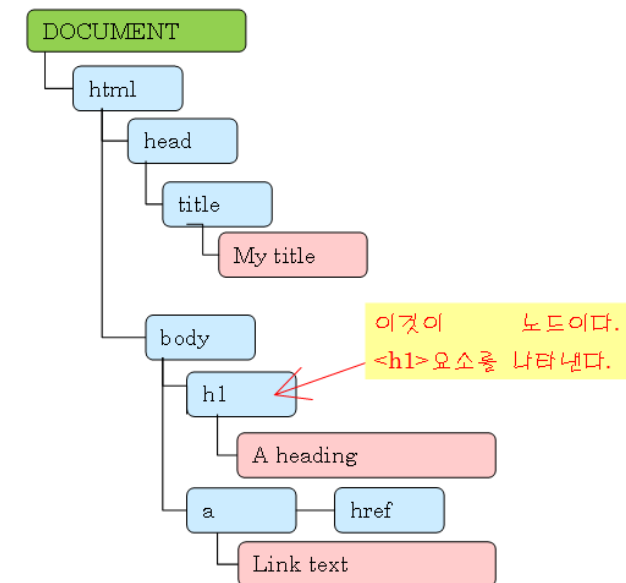
문서 객체 모델(DOM)

● DOM(Document Object Model)

- HTML 문서의 계층적인 구조를 트리(tree)로 표현
- HTML 구성하는 모든 요소들을 요소(element), 속성(attribute), 문자(text)로 구분
 - 태그 하나하나가 요소(element)에 해당
 - 태그 내의 속성 하나하나가 속성(attribute)에 해당
 - 태그로 감싸진 문자열 하나 하나가 문자(text)에 해당

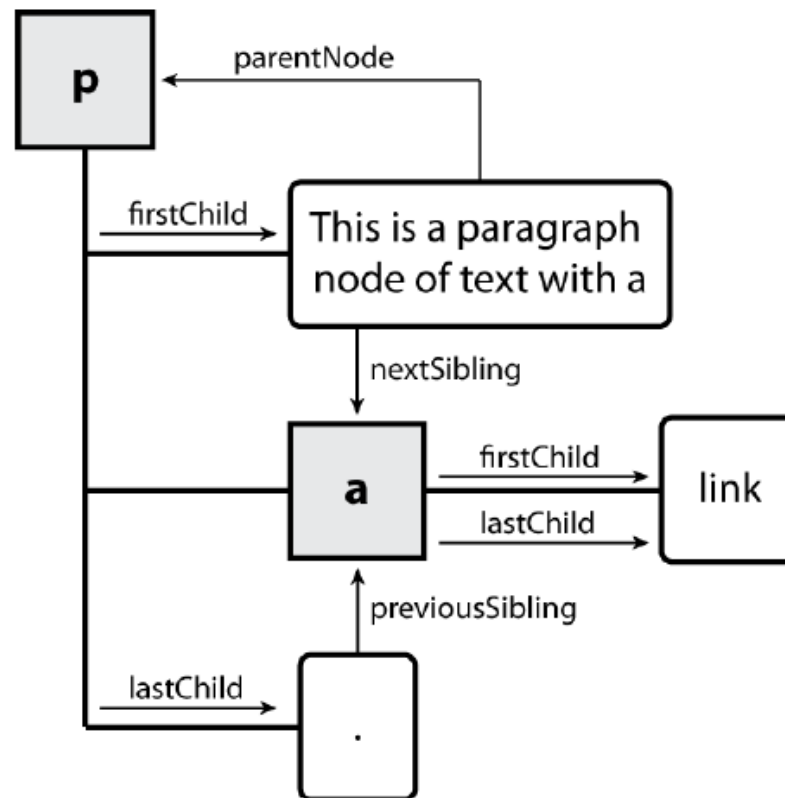
```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href="#" >Link text</a>
  </body>
</html>
```

DOM



HTML 요소 관계

```
<p id="foo">This is a paragraph of text with a  
<a href="/path/to/another/page.html">link</a>.</p>
```





● DOM을 바탕으로 무엇을 할 수 있는가?

- JavaScript can **change all the HTML elements** in the page
- JavaScript can **change all the HTML attributes** in the page
- JavaScript can **change all the CSS styles** in the page
- JavaScript can **remove existing HTML elements and attributes**
- JavaScript can **add new HTML elements and attributes**
- JavaScript can **react to all existing HTML events** in the page
- JavaScript can **create new HTML events** in the page

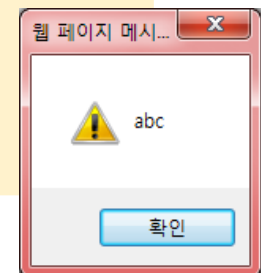
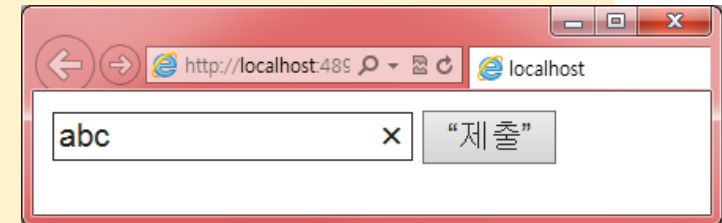
HTML 요소 찾기



Method	Description
<code>document.getElementById(<i>id</i>)</code>	Find an element by element id
<code>document.getElementsByTagName(<i>name</i>)</code>	Find elements by tag name
<code>document.getElementsByClassName(<i>name</i>)</code>	Find elements by class name

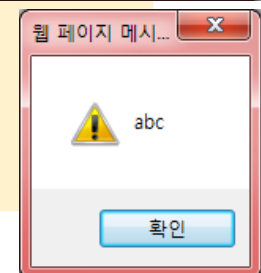
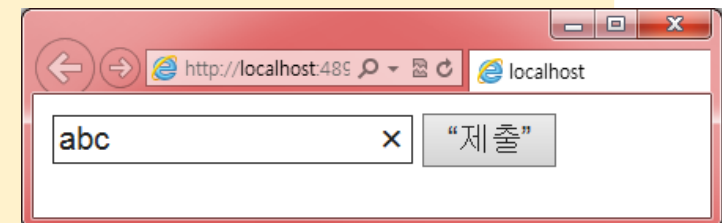
id로 HTML 요소 찾기

```
<!DOCTYPE >
<html>
<head>
  <script>
    function process() {
      var obj = document.getElementById("target");
      alert(obj.value);
    }
  </script>
</head>
<body>
  <form name="myform">
    <input type="text" id="target" name="text1">
    <input type="submit" value="제출" onclick="process()">
  </form>
</body>
</html>
```



name으로 HTML 요소 찾기

```
<!DOCTYPE >
<html>
<head>
  <script>
    function process() {
      var obj = document.myform.text1; // 또는 document.myform["text1"]
      alert(obj.value);
    }
  </script>
</head>
<body>
  <form name="myform">
    <input type="text" id="target" name="text1">
    <input type="submit" value="제출" onclick="process()">
  </form>
</body>
</html>
```

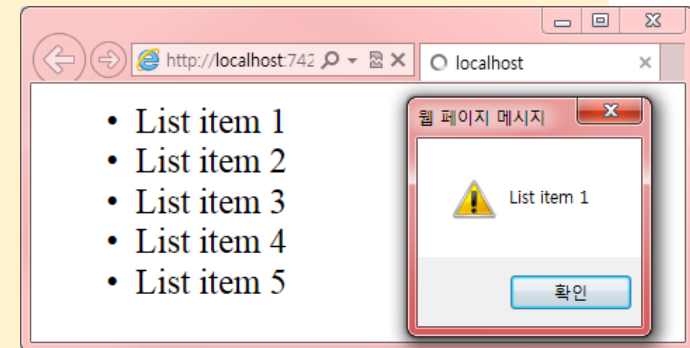


태그 이름으로 HTML 요소 찾기

```
<!DOCTYPE >
<html>
<body>
  <ul>
    <li>List item 1</li>
    <li>List item 2</li>
    <li>List item 3</li>
    <li>List item 4</li>
    <li>List item 5</li>

  </ul>
  <script>
    var list = document.getElementsByTagName('ul')[0];
    var allItems = list.getElementsByTagName('li');

    for (var i = 0, length = allItems.length; i < length; i++) {
      alert(allItems[i].firstChild.data);      //또는 alert(allItems[i].innerHTML);
    }
  </script>
</body>
</html>
```



HTML 요소의 내용 변경



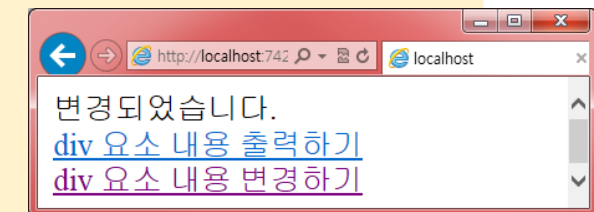
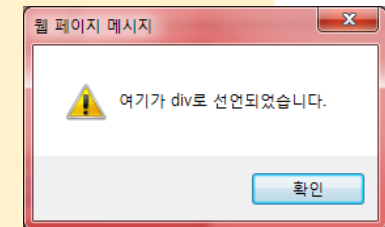
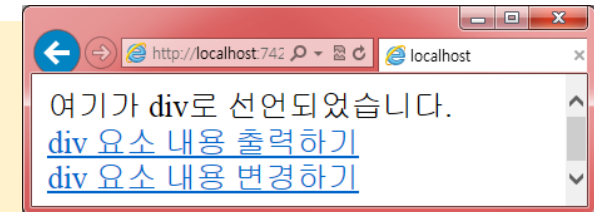
Method	Description
<code>element.innerHTML = new html content</code>	Change the inner HTML of an element
<code>element.attribute = new value</code>	Change the attribute value of an HTML element
<code>element.setAttribute(attribute, value)</code>	Change the attribute value of an HTML element
<code>element.style.property = new style</code>	Change the style of an HTML element

HTML 요소의 내용 변경

```
<!DOCTYPE html>
<html>

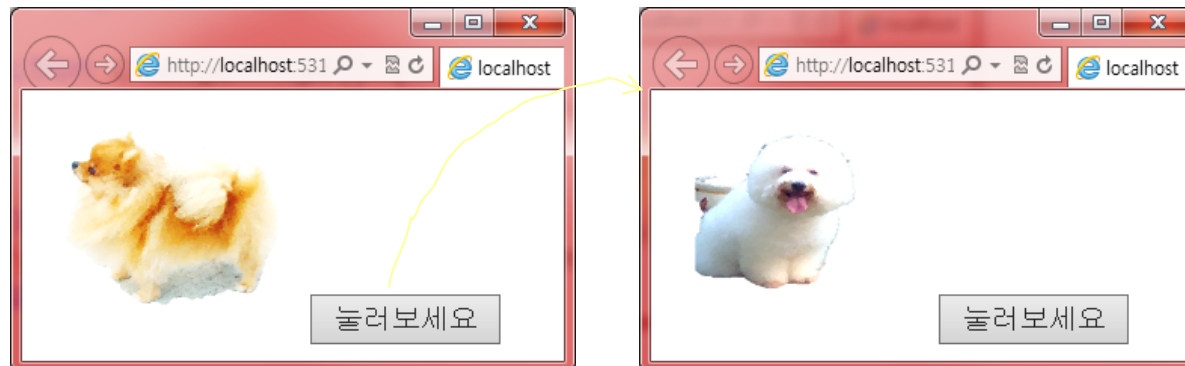
<head>
  <title></title>
  <script>
    function get() {
      var val = document.getElementById("ex").innerHTML;
      alert(val);
    }

    function set(v) {
      document.getElementById("ex").innerHTML = v;
    }
  </script>
</head>
<body>
  <div id="ex">여기가 div로 선언되었습니다.</div>
  <a href="#" onclick="get()">div 요소 내용 출력하기</a><br />
  <a href="#" onclick="set('변경되었습니다.')">div 요소 내용 변경하기</a>
</body>
</html>
```



요소의 속성 변경하기

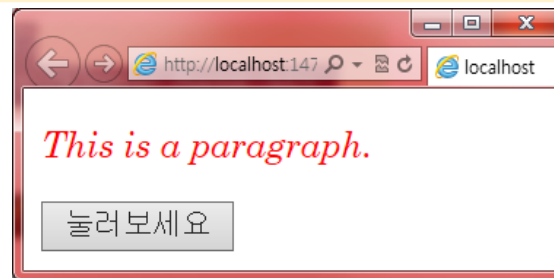
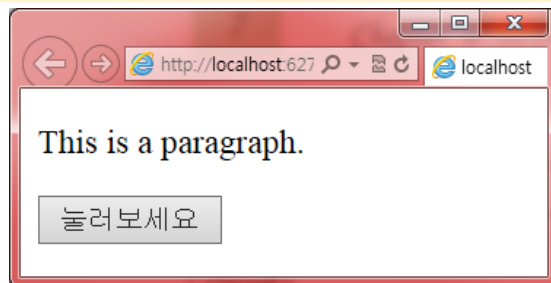
```
<!DOCTYPE html>
<html>
<head>
  <script>
    function changelImage() {
      document.getElementById("image").src = "poodle.png";
    }
  </script>
</head>
<body>
  
  <input type="button" onclick="changelImage()" value="눌러보세요" />
</body>
</html>
```



요소의 스타일 변경하기

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function changeStyle() {
      document.getElementById("p1").style.color = "red";
      document.getElementById("p1").style.fontFamily = "Century Schoolbook";
      document.getElementById("p1").style.fontStyle = "italic";
    }
  </script>

</head>
<body>
<p id="p1">This is a paragraph.</p>
<input type="button" onclick="changeStyle()" value="눌러보세요" />
</body>
</html>
```



Common DOM styling errors

- 스타일 속성 변경 시, 반드시 .style 사용해야 함

```
var clickMe = document.getElementById("clickme");  
clickMe.color = "red";  
clickMe.style.color = "red";
```

- 스타일 속성은 '.'이 아니고 대문자 사용

```
clickMe.style.font-size = "14pt";  
clickMe.style.fontSize = "14pt";
```

- 속성에 할당하는 값은 반드시 문자열 (" ")이어야 하고, 단위가 반드시 있어야 함

```
clickMe.style.width = 200;  
clickMe.style.width = "200px";  
clickMe.style.padding = "0.5em";
```

HTML 요소 추가 및 삭제

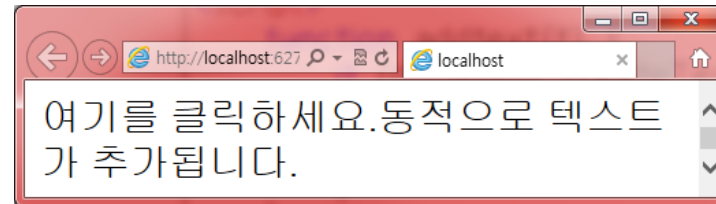
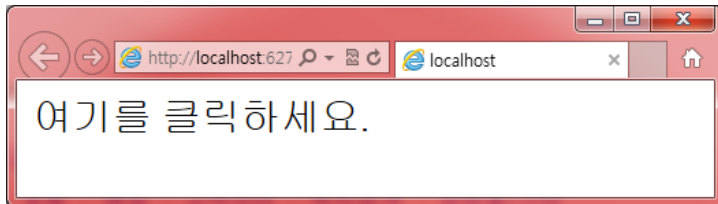


Method	Description
<code>document.createElement(<i>element</i>)</code>	Create an HTML element
<code>document.removeChild(<i>element</i>)</code>	Remove an HTML element
<code>document.appendChild(<i>element</i>)</code>	Add an HTML element
<code>document.replaceChild(<i>element</i>)</code>	Replace an HTML element
<code>document.write(<i>text</i>)</code>	Write into the HTML output stream

새로운 HTML 요소 생성

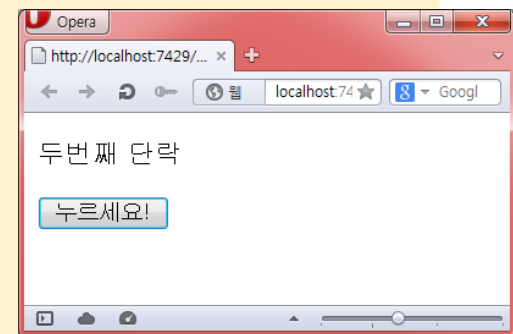
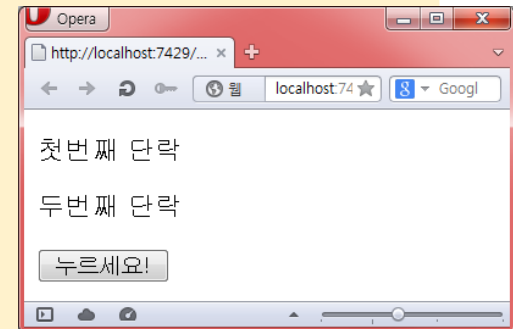
```
<script>  
  function addtext(t) {  
    var node = document.createTextNode(t);  
    document.getElementById("target").appendChild(node);  
  }  
</script>
```

```
<div id="target" onclick="addtext('동적으로 텍스트가 추가됩니다.')"  
  style="font: 20px bold;">여기를 클릭하세요.</div>
```



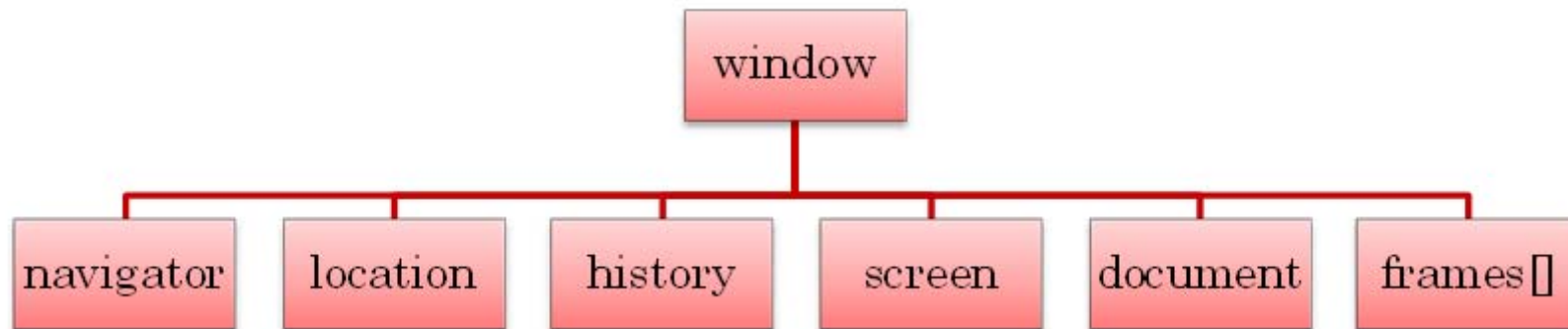
HTML 요소 삭제

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      function removeNode() {
        var parent = document.getElementById("target");
        var child = document.getElementById("p1");
        parent.removeChild(child);
      }
    </script>
  </head>
  <body>
    <div id="target">
      <p id="p1">첫번째 단락</p>
      <p id="p2">두번째 단락</p>
    </div>
    <button onclick="removeNode()">누르세요!</button>
  </body>
</html>
```



브라우저 객체 모델

- 웹 브라우저가 가지고 있는 모든 객체를 의미
- 최상위 객체는 window
 - 그 아래 navigator, location, history, screen, document, frames 객체



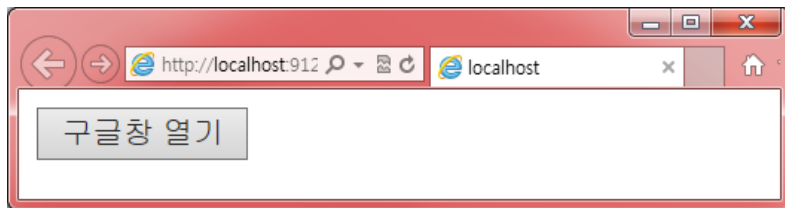
window 객체



- 브라우저의 하나의 윈도우를 표현하는 객체
 - 모든 JS's global objects, functions, variables는 window 객체의 멤버임
 - Global functions은 window 객체의 method
 - Global variables는 window 객체의 property
- window methods
 - window.open()
 - window.close()
 - window.moveTo()
 - window.resizeTo()
 - window.onload()
 - window.unload()

새로운 윈도우 오픈

```
<!DOCTYPE html>
<html>
<head></head>
<body>
  <form>
    <input type="button" value="구글창 열기"
    onclick="window.open('http://www.google.com', '_blank', 'width=300, height=300', true)">
  </form>
</body>
</html>
```



screen 객체



● window.screen

- 사용자 모니터 화면에 대한 정보를 포함하는 객체
- window 생략 가능
- Property

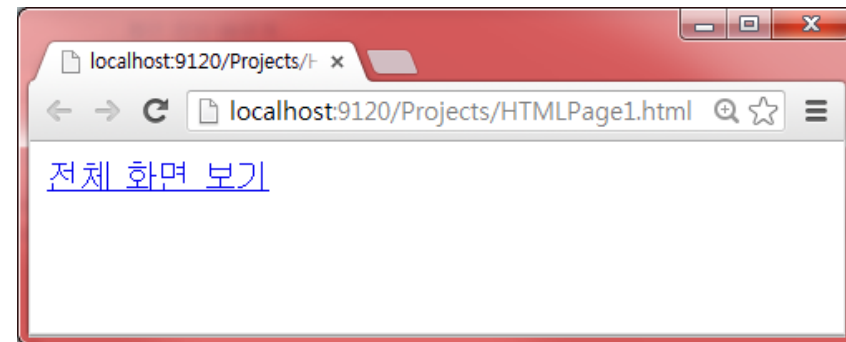
속성	설명
<u>availHeight</u>	화면의 높이를 반환(윈도우에서 <u>태스크바</u> 를 제외한 영역)
<u>availWidth</u>	화면의 너비를 반환(윈도우에서 <u>태스크바</u> 를 제외한 영역)
<u>colorDepth</u>	컬러 팔레트의 비트 깊이를 반환
<u>height</u>	화면의 전체 높이를 반환
<u>pixelDepth</u>	화면의 컬러 해상도(bits per pixel)를 반환
<u>width</u>	화면의 전체 너비를 반환

- screen.colorDepth
 - 한 가지 색을 표현하기 위해 사용된 비트 수 (24bit or 32bit)

예제

```
<script>
  function maxopen(url, winattributes) {
    var maxwindow = window.open(url, "", winattributes)
    maxwindow.moveTo(0, 0);
    maxwindow.resizeTo(screen.availWidth, screen.availHeight)
  }

</script>
<a href="#" onClick="maxopen('http://www.google.com', 'resize=1, scrollbars=1, status=1');
return false">전체 화면 보기</a>
```



location 객체

● window.location

- 현재 페이지 주소(URL)에 대한 정보 및 새로운 페이지로 이동 (redirect)하기 위한 정보 포함

- Property

속성	설명
<u>hash</u>	URL 중에서 앵커 부분을 반환한다. (#section1과 같은 부분)
<u>host</u>	URL 중에서 <u>hostname</u> 와 <u>port</u> 를 반환
<u>hostname</u>	URL 중에서 <u>hostname</u> 을 반환
<u>href</u>	전체 URL을 반환
<u>pathname</u>	URL 중에서 경로(path)를 반환
<u>port</u>	URL 중에서 <u>port</u> 를 반환
<u>protocol</u>	URL 중에서 <u>protocol</u> 부분을 반환
<u>search</u>	URL 중에서 쿼리(query) 부분을 반환

- Methods (for url redirect)

- 현재 윈도우에 새 페이지 로드
- location.assign()
 - History 유지, back 버튼을 통해서 이전 페이지 복귀 가능
- location.replace()
 - History 삭제, back 버튼으로 이전 페이지 복귀 불가

예제



```
<html>
<head>
<script>
function newDoc() {
    window.location.assign("https://www.konkuk.ac.kr");
}
</script>
</head>
<body>

<input type="button" value="Load new document" onclick="newDoc()">

</body>
</html>
```


navigator 객체

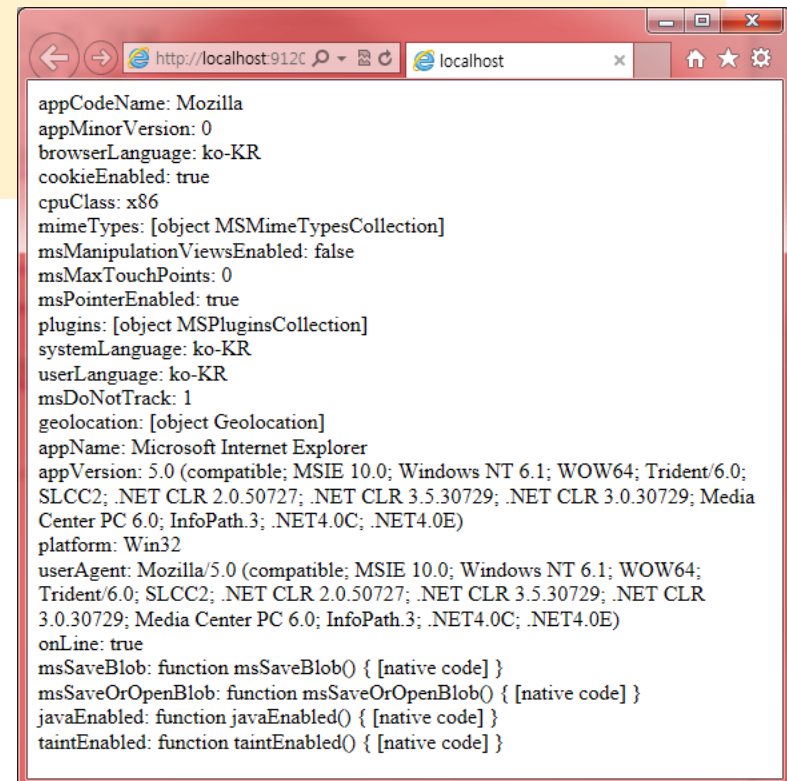
- window.navigator
 - 사용자의 브라우저에 대한 정보 포함
 - Property

속성	설명
<u>appCodeName</u>	브라우저의 코드 네임
<u>appName</u>	브라우저의 이름
<u>appVersion</u>	브라우저의 버전 정보
<u>cookieEnabled</u>	브라우저에서 쿠키가 활성화되어 있는지 여부
<u>onLine</u>	브라우저가 인터넷에 연결되어 있으면 true
<u>platform</u>	브라우저가 컴파일된 플랫폼
<u>userAgent</u>	브라우저에서 서버로 가는 user-agent 헤더

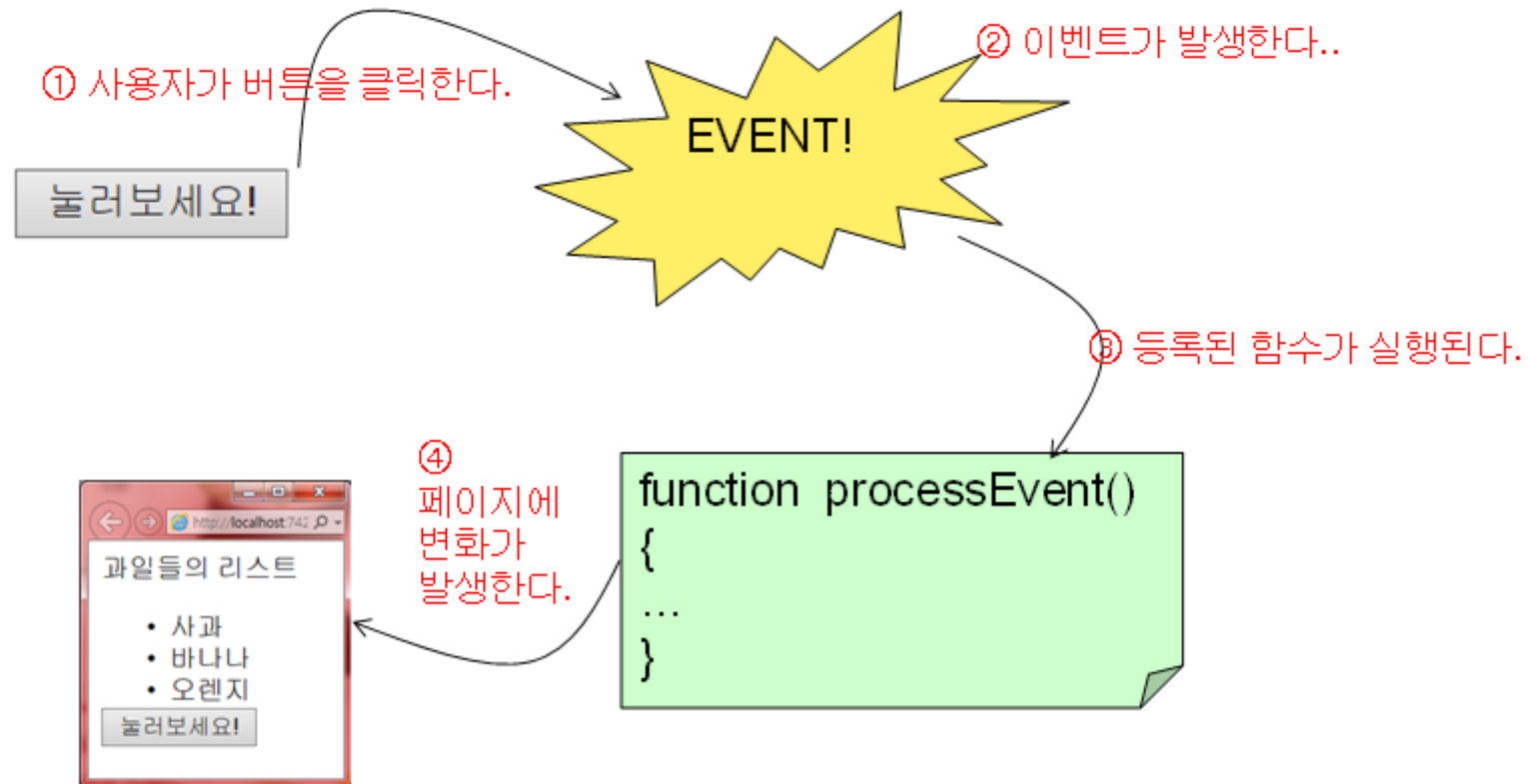
- appCodeName = "Mozilla"
 - Chrome, Firefox, IE, Safari, Opera의 application code name
- appName = "Netscape"
 - IE11, Chrome, Firefox, Safari의 application name

예제

```
<html>
  <body>
    <script>
      for (var key in navigator) {
        value = navigator[key];
        document.write(key + ": " + value + "<br>");
      }
    </script>
  </body>
</html>
```



이벤트 처리



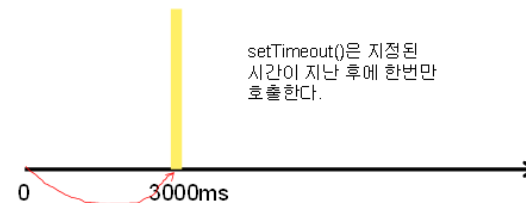
Timing 이벤트

- 주어진 시간 주기에 따라 코드가 실행되도록 허용
- 주요 methods
 - `window.setTimeout(function, milliseconds)`
 - 명시된 milliseconds 후에, function 실행

setTimeout(code, millisec)

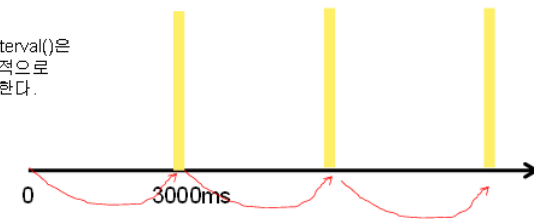
호출되는 함수의 이름, 호출되는 함수를 여기서 직접 정의할 수도 있다.

함수를 호출하기 전에 불러야 하는 시간



- `window.setInterval(function, milliseconds)`
 - 명시된 milliseconds 단위로 코드가 반복 실행

`setInterval()`은 주기적으로 호출한다.



- window 생략 가능, 보통 생략하고 사용**

Timing 이벤트



● 주요 methods

- `window.clearTimeout(timeoutVariable)`
 - `setTimeout()` 함수의 실행을 멈춤

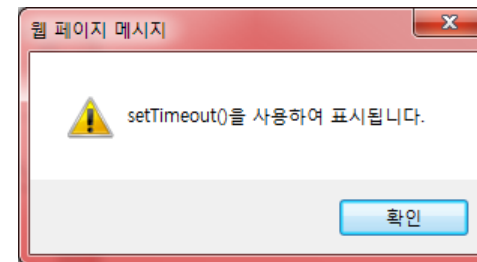
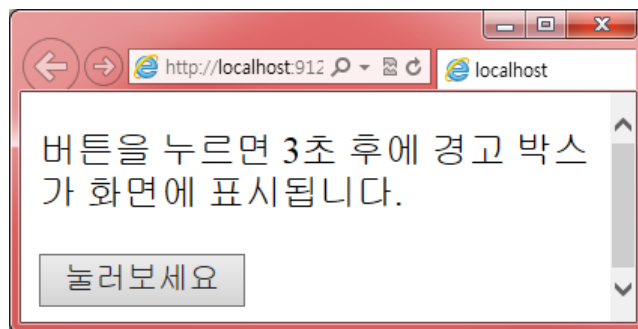
```
myVar = setTimeout(function, milliseconds);  
clearTimeout(myVar);
```

- `window.clearInterval(timerVariable)`
 - `setInterval()` 함수의 실행을 멈춤

```
myVar = setInterval(function, milliseconds);  
clearInterval(myVar);
```

예제

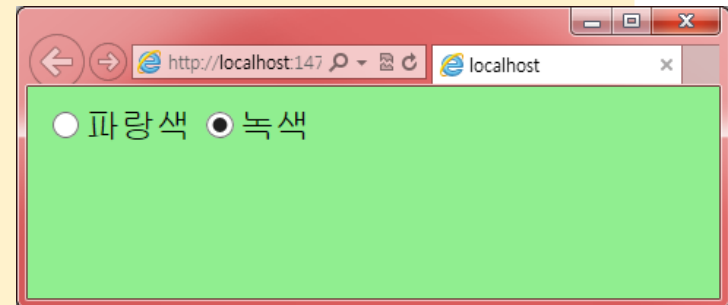
```
<!DOCTYPE html>
<html>
<head>
  <script >
    function showAlert() {
      setTimeout(function () { alert("setTimeout()을 사용하여 표시됩니다.") }, 3000);
    }
  </script>
</head>
<body>
  <p>버튼을 누르면 3초 후에 경고 박스가 화면에 표시됩니다. </p>
  <button id="button1" onclick="showAlert()">눌러보세요</button>
</body>
</html>
```



onclick 이벤트

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function changeColor(c) {
      document.getElementById("target").style.backgroundColor = c;
    }
  </script>
</head>
<body id="target">

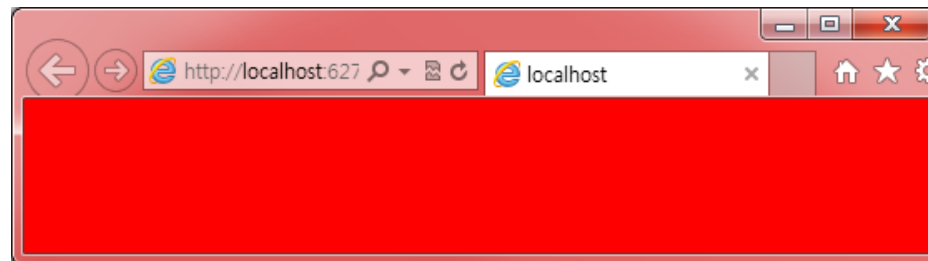
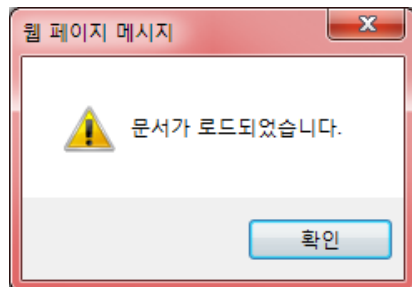
  <form method="POST">
    <input type="radio" name="C1" value="v1"
      onclick="changeColor('lightblue')">파랑색
    <input type="radio" name="C1" value="v2"
      onclick="changeColor('lightgreen')">녹색
  </form>
</body>
</html>
```



onload / onunload 이벤트

- 사용자가 웹 페이지에 진입하거나 나갈 때 발생하는 이벤트
 - 주로 <body> 태그와 함께 사용

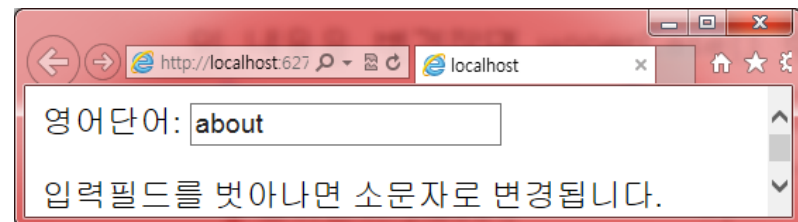
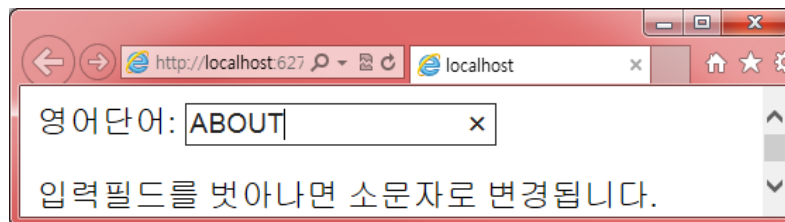
```
<html>
<head>
  <script>
    function onLoadDoc() {
      alert("문서가 로드되었습니다.");
      document.body.style.backgroundColor = "red";
    }
  </script>
</head>
<body onload="onLoadDoc();">
</body>
</html>
```



onchange 이벤트

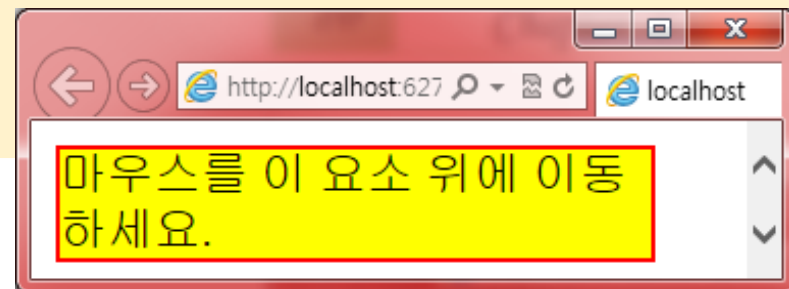
- 주로 input 필드와 함께 사용됨
 - focus 상태였다가 벗어나면 onchange 이벤트 발생

```
<!DOCTYPE html>
<html>
<head>
<script>
  function sub() {
    var x = document.getElementById("name");
    x.value = x.value.toLowerCase();
  }
</script>
</head>
<body>
영어단어: <input type="text" id="name" onchange="sub()">
<p>입력필드를 벗어나면 소문자로 변경됩니다.</p>
</body>
</html>
```



onmouseover / onmouseout 이벤트

```
<html>
<head>
  <script>
    function OnMouseIn(elem)
    {
      elem.style.border = "2px solid red";
    }
    function OnMouseOut(elem)
    {
      elem.style.border = "";
    }
  </script>
</head>
<body>
  <div style="background-color: yellow; width: 200px"
    onmouseover="OnMouseIn (this)" onmouseout="OnMouseOut (this)">
    마우스를 이 요소 위에 이동하세요.
  </div>
</body>
</html>
```

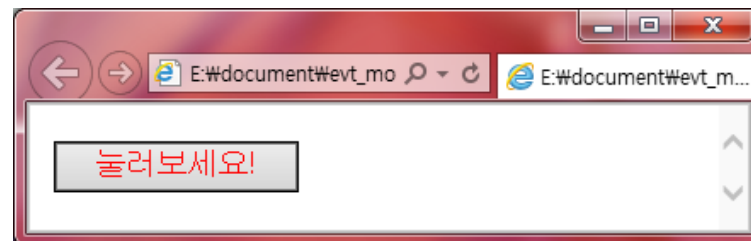
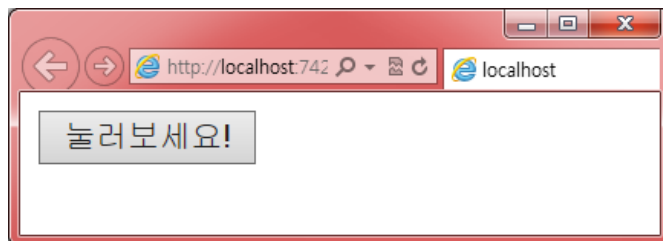


onmousedown / onmouseup 이벤트

```
<html>
<head>
  <script>
    function OnButtonDown(button) {

        button.style.color = "#ff0000";

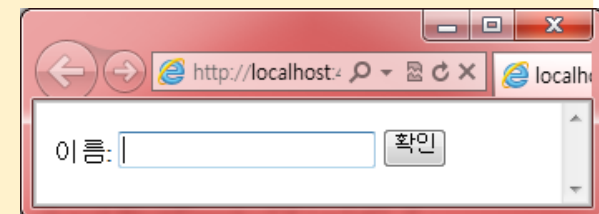
    }
    function OnButtonUp(button) {
        button.style.color = "#000000";
    }
  </script>
</head>
<body>
  <button onmousedown="OnButtonDown (this)" onmouseup="OnButtonUp
(this)">눌러보세요!</button>
</body>
```



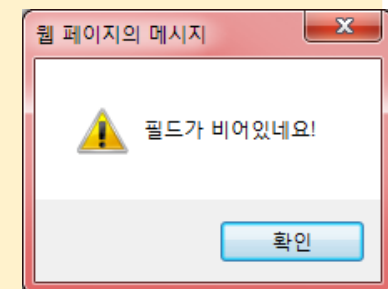
폼의 유효성 검증

- 입력 필드에서의 잘못을 검증하는 작업
- 공백 및 입력 값 길이 검증
 - length 속성 활용

```
<script>
function checkNotEmpty(field) {
    if (field.value.length == 0) {
        alert("필드가 비어있네요!");
        field.focus();
        return false;
    }
    return true;
}
</script>
<form>
    이름: <input type='text' id='user' />
    <input type='button'
        onclick="checkNotEmpty(document.getElementById('user'))"
        value='확인' />
</form>
```



A screenshot of a web browser window. The address bar shows 'http://localhost:'. Below the address bar, there is a form with a text input field labeled '이름:' and a button labeled '확인'.



입력값 검증



- 정규식(regular expression)
 - 특정한 규칙을 가지고 있는 문자열들을 표현하는 수식
 - 문법: `/pattern/modifiers;`
 - ex) `/^[0-9]+abc$/`, `/JavaScript/i`

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Expression	Description
[<u>abc</u>]	Find any character between the brackets
[<u>^</u> abc]	Find any character NOT between the brackets
[<u>0-9</u>]	Find any character between the brackets (any digit)
[<u>^</u> 0-9]	Find any character NOT between the brackets (any non-digit)
(<u>x</u> <u>y</u>)	Find any of the alternatives specified

입력값 검증



Quantifier	Description
<u>$n\pm$</u>	Matches any string that contains at least one n
<u>n^*</u>	Matches any string that contains zero or more occurrences of n
<u>$n?$</u>	Matches any string that contains zero or one occurrences of n
<u>$n\{X\}$</u>	Matches any string that contains a sequence of X n 's
<u>$n\{X,Y\}$</u>	Matches any string that contains a sequence of X to Y n 's
<u>$n\{X, \}$</u>	Matches any string that contains a sequence of at least X n 's
<u>$n\\$</u>	Matches any string with n at the end of it
<u>n</u>	Matches any string with n at the beginning of it
<u>$?=n$</u>	Matches any string that is followed by a specific string n
<u>$?!n$</u>	Matches any string that is not followed by a specific string n

입력값 검증



● 정규식 methods

● exec()

- 주어진 문자열에서 정규식에 match 하는 패턴이 있는지 검사하고 매치되는 첫 번째 문자열 반환
- 정규식.exec('문자열')

● test()

- 주어진 문자열에서 정규식에 match되는 패턴이 있는지 검사하고, true / false 반환
- 정규식.test('문자열')

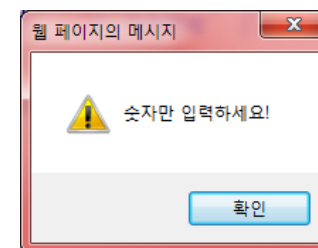
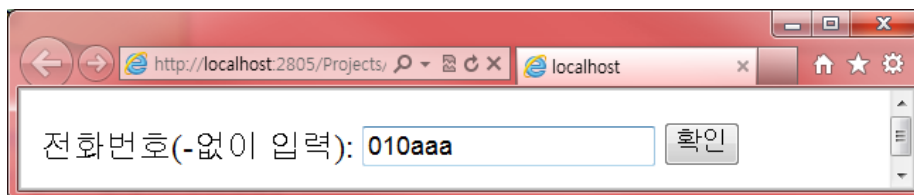
● 문자열 검증 method

● match()

- 기능은 exec()와 유사
- string.match(정규식)

숫자 검증 예제

```
<script>
function checkNumeric(elem) {
    var exp = /^[0-9]+$/;
    if (elem.value.match(exp)) {           // 또는 if(exp.exec(elem.value))
        return true;
    } else {
        alert('숫자만 입력하세요!');
        elem.focus();
        return false;
    }
}
</script>
<form>
전화번호(-없이 입력): <input type='text' id='phone' />
<input type='button'
    onclick="checkNumeric(document.getElementById('phone'))"
    value='확인' />
</form>
```



이벤트 핸들러를 통한 이벤트 처리

● Unobtrusive JS

- 눈에 거슬리지 않는, 겸손한
- HTML with minimal JS inside
- Use DOM to attach and execute all JS functions
- Separates the page into three parts:
 - **contents (HTML files)** – what is it?
 - **presentation /style (CSS files)** – how does it look?
 - **code/scripts (JS files)** – how does it behave or respond to user interaction?

Obtrusive event handlers (bad)

- 이벤트 핸들러
 - 이벤트 발생 시 수행해야 할 동작을 정의
- Obtrusive event handler
 - HTML 문서 내에 이벤트 발생에 따른 이벤트 핸들러를 구체적으로 명시하는 방식

```
<button id="ok" onclick="okayClick();" >OK</button>
```

html

```
// called when OK button is clicked  
function okayClick() {  
    alert("booyah");  
}
```

JS

OK

Output

Goal: remove all JS codes (onclick = "okayClick();") form HTML code

Attaching an event handler in JS

```
// where element is a DOM element object  
element.event = function;
```

```
var button1 = document.getElementById("ok");  
button1.onclick = okayClick;
```

- id=ok 인 DOM 객체의 onclick 이벤트에 okayClick 핸들러 추가
- 단, 핸들러 추가 시, **괄호 () 사용하지는 안됨!**

Attaching an event handler in JS

< HTML >

```
<head>
    <script src="myfile.js" type="text/javascript"></script>
</head>
<body>
    <div><button id="ok">OK</button></div>
```

<myfile.js>

```
var button1 = document.getElementById("ok");
button1.onclick = okayClick;           // error: button1 is null
```

● JS 코드 실행 시점

- 브라우저가 script 태그를 로딩할 때 실행
 - 스크립트 코드가 <head></head> 영역에서 로딩
 - 현 시점에서 브라우저는 아직 <body> 태그 영역을 로딩하지 않았음
 - 따라서, JS 코드 내에 있는 DOM object들은 아직 생성되지 않았음

이벤트 핸들러는 페이지 로딩이 끝난 다음에 추가되어야 함!!

인터넷 프로그래밍

Attaching an event handler in JS

● **window.onload** 이벤트

- 브라우저가 페이지 로딩을 마쳤을 때 발생하는 이벤트
- 다른 모든 이벤트 핸들러는 window.onload 핸들러에 추가하여 생성

```
// this will run once the page has finished loading
function functionName() {
    element.event = functionName;
    element.event = functionName;
    ...
}
window.onload = functionName;           // global code
```

<myfile.js>

```
// called when page loads; sets up event handlers
window.onload = pageLoad;           // global code
function pageLoad() {
    var button1 = document.getElementById("ok");
    button1.onclick = okayClick; }

function okayClick() {
    alert("booyah"); }
```

Common Errors

- 이벤트 핸들러 다음에는 ()를 붙여서는 안됨!
 - 핸들러는 함수 호출이 아니라 이벤트 발생 시 동작할 함수의 정의를 할당하는 것임

```
window.onload = pageLoad();  
window.onload = pageLoad;  
okButton.onclick = okayClick();  
okButton.onclick = okayClick;
```

- 이벤트 이름은 모두 소문자여야 함!

```
window.onLoad = pageLoad;  
window.onload = pageLoad;
```

Anonymous function

- 핸들러로 anonymous function 사용 가능
 - 함수 이름 없이 함수 정의
 - 변수에 저장될 수 있고, 핸들러로 추가될 수 있음

```
window.onload = function() {  
    var okButton = document.getElementById("ok");  
    okButton.onclick = okayClick;  
};  
  
function okayClick() {  
    alert("booyah");  
}
```

예제

```
<!DOCTYPE html>
<html>
<head><script src="myfile.js" type="text/javascript"></script></head>
<body>
  <p>버튼을 누르면 3초 후에 경고 박스가 화면에 표시됩니다. </p>
  <button id="button1">눌러보세요</button>
</body>
</html>
```

```
window.onload = pageLoad;
```

```
function pageLoad() {
  var b1 = document.getElementById("button1");
  b1.onclick = showAlert; }
```

```
function showAlert() {
  setTimeout(function () { alert("setTimeout()을 사용하여 표시됩니다."); }, 3000);
}
```

