

웹프로그래밍

2024-1

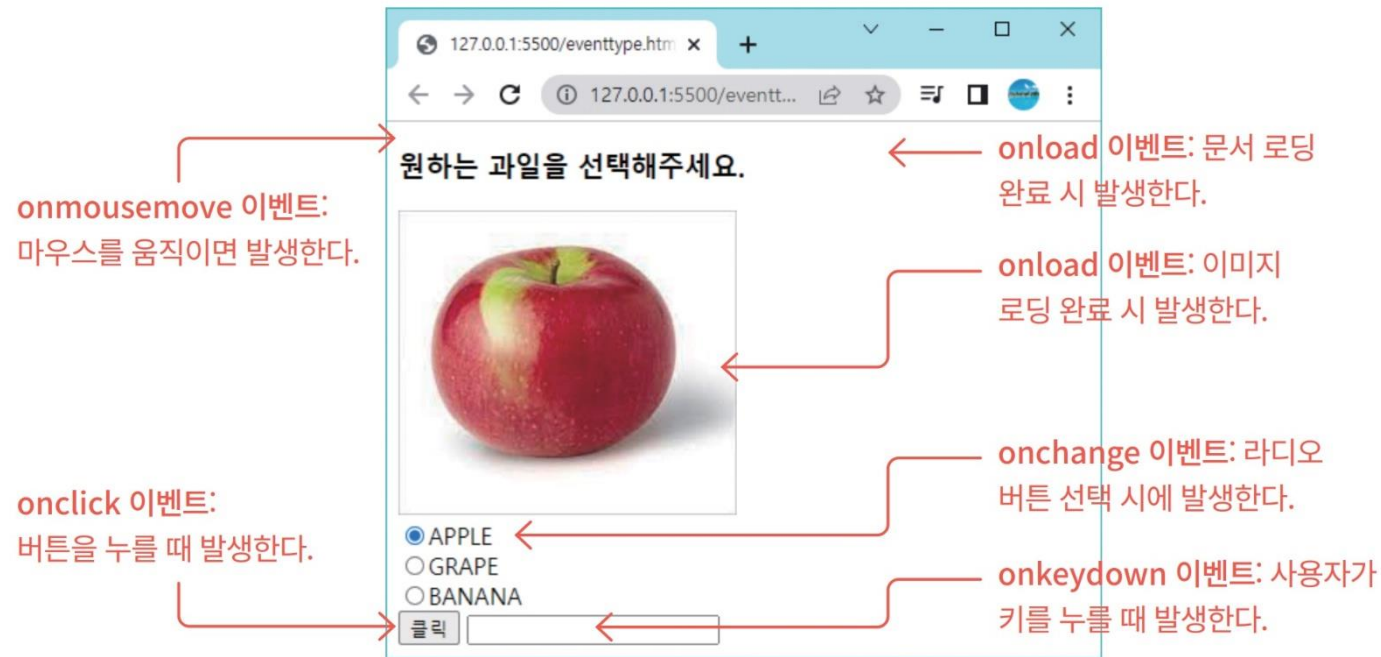
11. 이벤트 처리, 입력 검증, 위치 정보, 웹 워커

이번 장의 목표

- HTML 요소를 클릭하였을 때 발생하는 이벤트를 처리할 수 있나요?
- 마우스를 요소 안으로 움직였을 때 발생하는 이벤트를 처리할 수 있나요?
- 사용자가 8글자 이상을 입력하였는지 검사할 수 있나요?
- 사용자가 숫자만 입력하였는지 검사할 수 있나요?

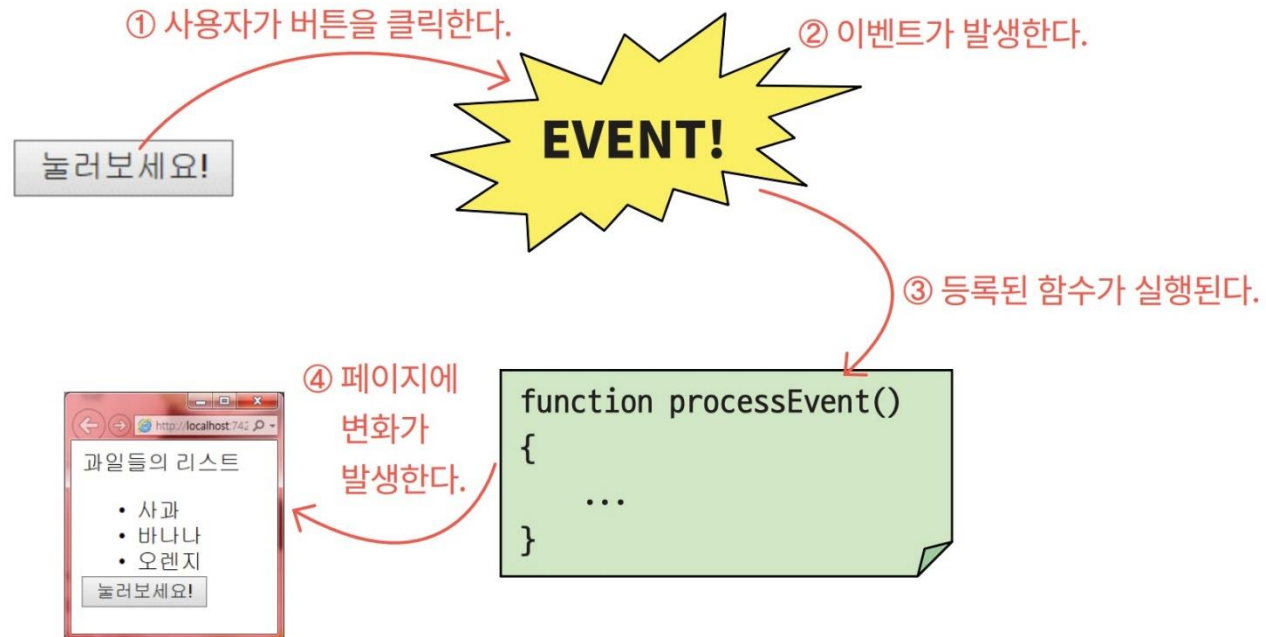
이벤트란 무엇인가?

- 웹 페이지에서 상호작용이 발생하면 이벤트가 일어난다.
- 예를 들어 사용자가 버튼을 클릭하거나, 특정 요소 위로 마우스 커서를 가져가면 이벤트가 발생함



이벤트 처리 절차

- 이벤트를 처리하는 코드를 버튼에 등록하여야 함.
- 이를 이벤트 핸들러, 또는 이벤트 리스너라고 함.



이벤트의 종류

- 마우스 관련 이벤트

이벤트	이벤트 리스너	설명
click	onclick	마우스로 요소를 클릭했을 때 발생한다.
mouseover	onmouseover	마우스 커서가 요소 위로 올 때 발생한다.
mouseout	onmouseout	마우스 커서가 요소를 떠날 때 발생한다.
mousedown	onmousedown	요소 위에서 마우스 버튼을 눌렀을 때 발생한다.
mouseup	onmouseup	요소 위에서 마우스 버튼을 놓을 때 발생한다.
mousemove	onmousemove	마우스 움직임이 있을 때 발생한다.

이벤트의 종류

• 키보드 입력 요소 이벤트

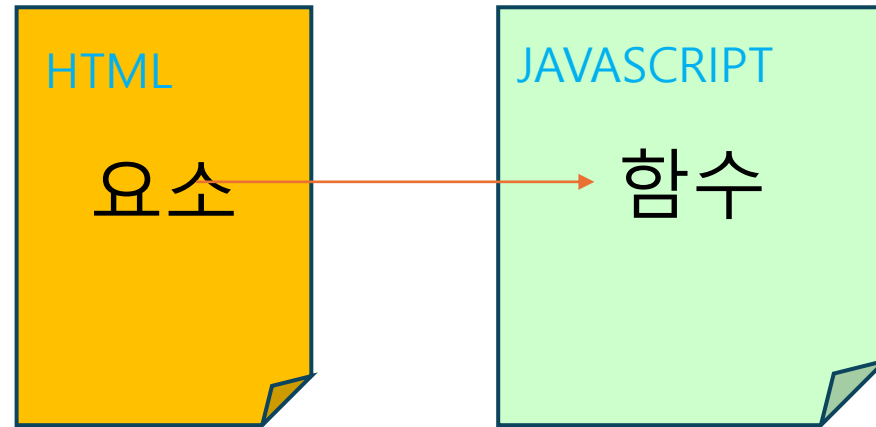
이벤트	이벤트 리스너	설명
keydown keyup	onkeydown onkeyup	사용자가 키를 눌렀다가 놓을 때 발생한다.

이벤트	이벤트 리스너	설명
focus	onfocus	사용자가 입력 요소에 키보드 입력을 시도할 때 발생한다.
submit	onsubmit	사용자가 입력 양식을 제출할 때 발생한다.
blur	onblur	포커스가 입력 요소를 벗어나는 경우에 발생한다.
change	onchange	사용자가 입력 요소의 값을 변경할 때 발생한다.

이벤트	이벤트 리스너	설명
load	onload	브라우저가 페이지 로드를 완료하면 발생한다.
unload	onunload	사용자가 현재 웹 페이지를 떠날 때 발생한다.
resize	onresize	사용자가 브라우저 윈도우의 크기를 조절할 때 발생한다.

이벤트 리스너 만들기

1. 인라인 방식: HTML 요소 안에 코드 추가
2. 객체의 이벤트 리스너 속성에 함수를 대입
3. 객체의 `addEventListener()` 함수를 사용
4. 익명 함수로 이벤트 리스너 작성



이벤트 리스너 만들기

1. 인라인 방식

- `<element event="자바스크립트 코드">`와 같이 HTML 요소에 코드를 작성해서 붙이는 방법.
- `<div onmouseover="this.style.backgroundColor='blue'"> Hello World! </div>`

2. 객체의 이벤트 리스너 속성에 함수를 대입

```
<div id="special"> Hello World! </div>
<script>
function changeColor() {
  document.getElementById("special").style.backgroundColor='blue';
}
document.getElementById("special").onmouseover = changeColor;
</script>
```


이벤트 리스너 만들기

3. 객체의 addEventListener() 함수를 사용

- DOM 객체가 가지고 있는 addEventListener() 함수를 이용하여 이벤트 리스너를 등록하는 방법.
- 객체의 이벤트 리스너 속성에 함수를 대입

```
let obj = document.getElementById("special");  
obj.addEventListener("mouseover", changeColor);
```

```
obj.addEventListener("mouseover", firstFunction);  
obj.addEventListener("mouseover", secondFunction);
```

이벤트 리스너 만들기

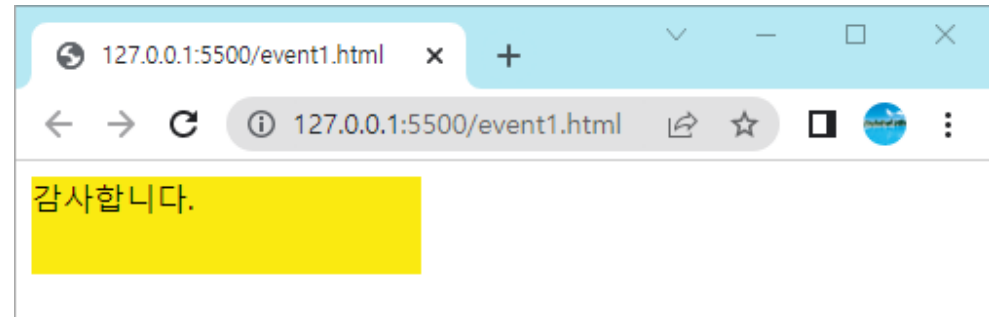
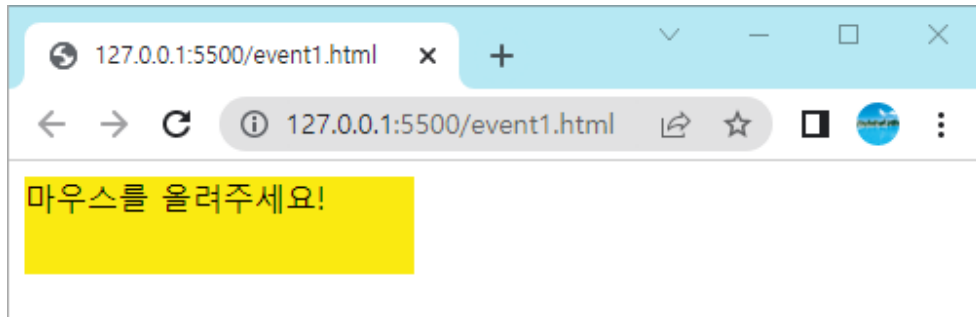
4. 익명 함수로 이벤트 리스너 작성

- 익명 함수(이름없는 함수)나 화살표 함수를 사용하여 이벤트 리스너를 작성할 수도 있음.

```
let obj = document.getElementById("special");  
obj.addEventListener("mouseover", function() {  
  document.getElementById("special").style.backgroundColor='blue';  
});
```

예제

- 요소에 마우스가 올라오면
요소의 내용을 변경하는 코드를 작성해보자.



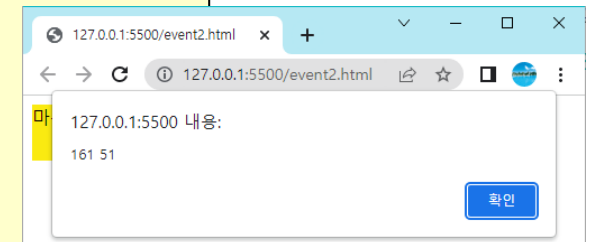
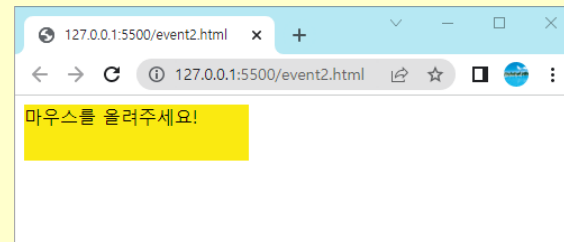
예제

```
<!DOCTYPE html>
<html>
<body>
  <div onmouseover="mouseOver(this)" onmouseout="mouseOut(this)"
    style="background-color:#faea11; width:200px; height:50px;">
    마우스를 올려주세요!</div>
  <script>
    function mouseOver(obj) {
      obj.innerHTML = "감사합니다."
    }
    function mouseOut(obj) {
      obj.innerHTML = "마우스를 올려주세요!"
    }
  </script>
</body>
</html>
```

이벤트 객체

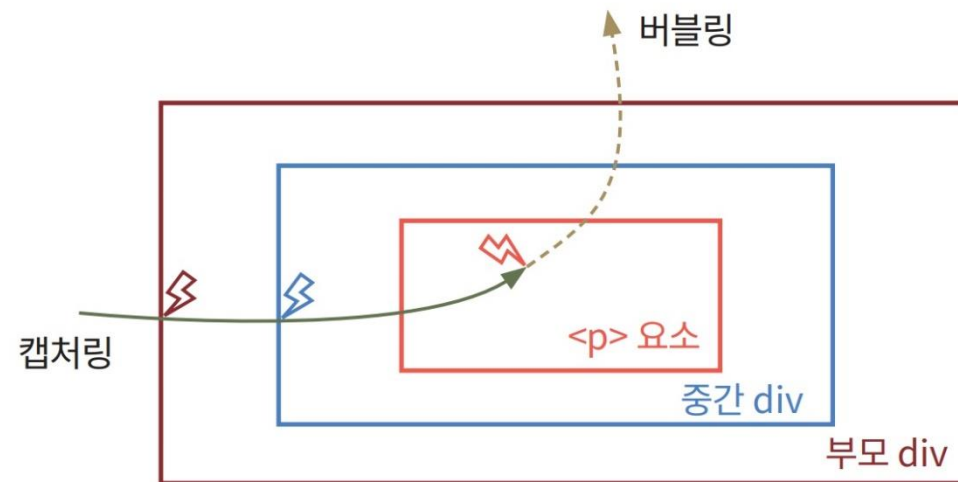
- 이벤트 리스너는 이벤트 객체(event object)를 전달받는다.
- 이벤트 객체는 이벤트마다 달라짐
 - 예를 들어 mouseover 이벤트의 경우, 마우스 좌표와 마우스 버튼 정보 등이 포함됨

```
<!DOCTYPE html>
<html>
<body>
  <div id="special" style="background-color:#faea11;width:200px;height:50px;">
    마우스를 올려주세요!</div>
  <script>
    let obj = document.getElementById("special");
    obj.onmouseover= function (e) {
      x=e.clientX;
      y=e.clientY;
      alert(x+" "+y); //좌표
      alert(e.type); // 이벤트
      alert(e.target); //이벤트 적용된 요소
    };
  </script>
</body>
</html>
```



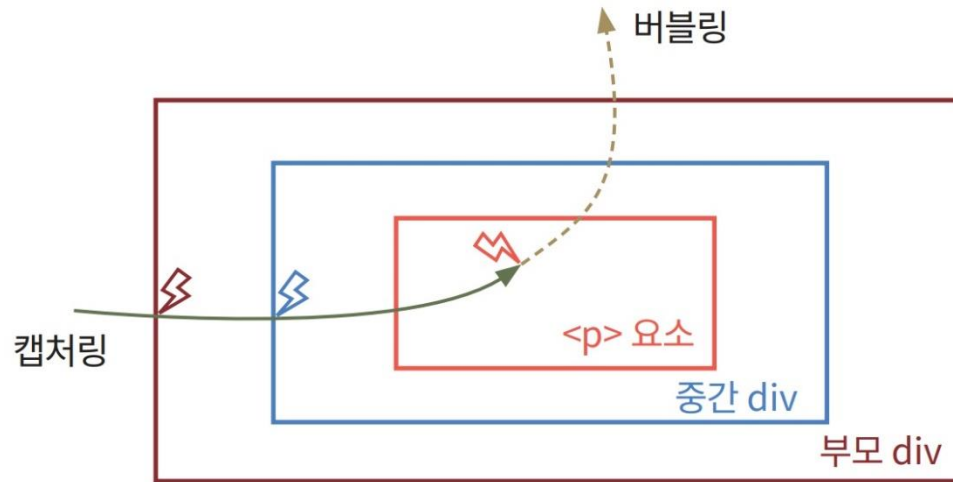
이벤트 전파

- 예를 들어서 <div> 요소 안에 <p> 요소가 있고, 사용자가 <p> 요소를 클릭하면 어떤 요소의 "onclick" 이벤트가 먼저 처리되어야 할까?



버블링(bubbling)과 캡처링(capturing)

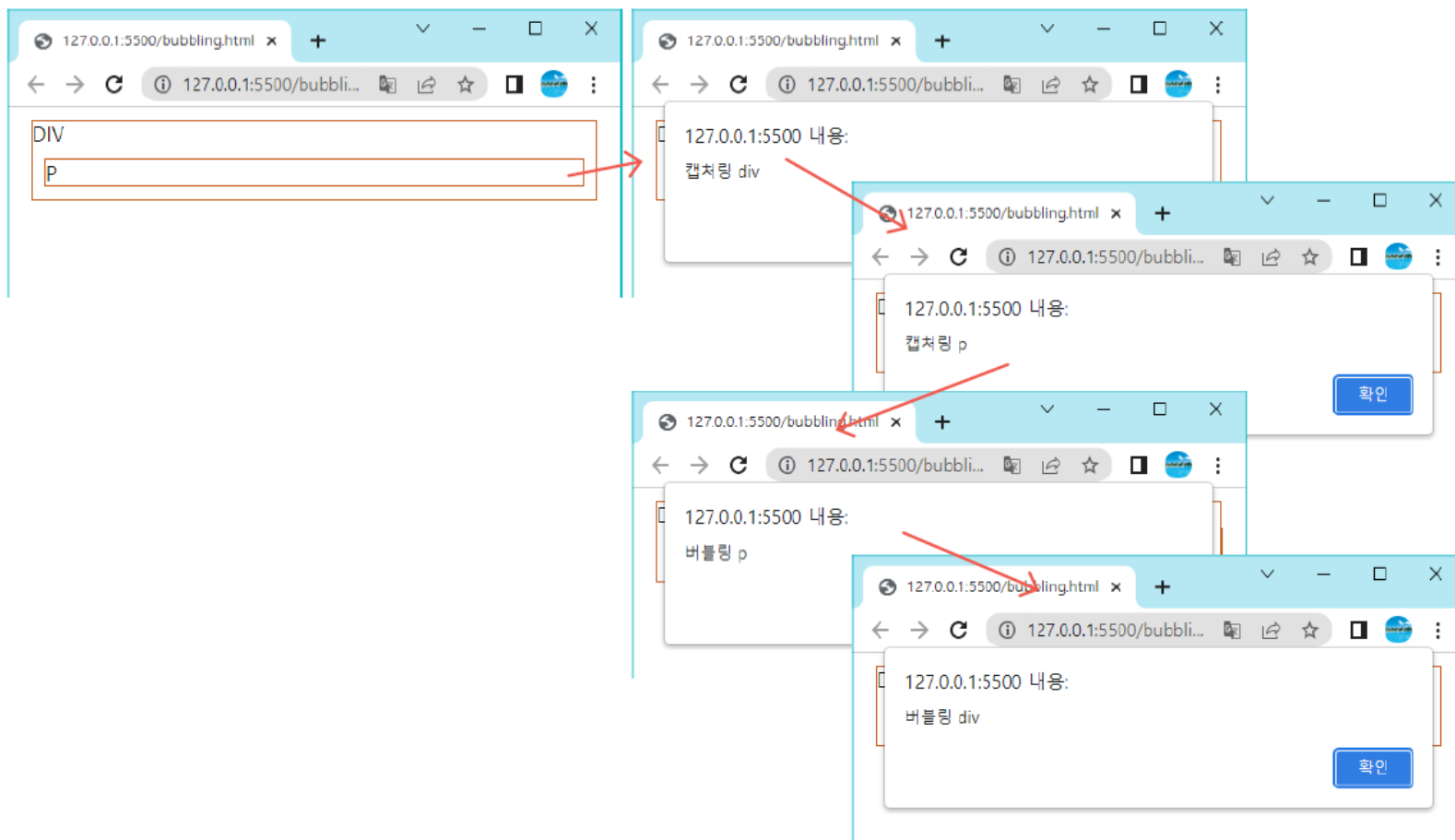
- 버블링은 가장 안쪽 요소의 이벤트가 먼저 처리된 다음, 바깥쪽 이벤트가 처리된다.
- 캡처링은 가장 바깥쪽 요소의 이벤트가 먼저 처리된 다음, 안쪽 요소의 이벤트가 처리된다.



```
addEventListener(event, function, useCapture);
```

true이면 캡처링이다(디폴트는 버블링) ←

예제



예제

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <style>
    p, div {
      margin: 10px;
      border: 1px solid rgb(199, 89, 25);
    }
  </style>
</head>
<body>
  <div id="div1">DIV
    <p id="p1">P</p>
  </div>
  <script>
    let elem1= document.getElementById("div1");
    elem1.addEventListener("click", function() {alert(`캡처링 div`);}, true);
    elem1.addEventListener("click", function() {alert(`버블링 div`);});
    elem2= document.getElementById("p1");
    elem2.addEventListener("click", function() {alert(`캡처링 p`);}, true);
    elem2.addEventListener("click", function() {alert(`버블링 p`);});
  </script>
</body>
</html>
```

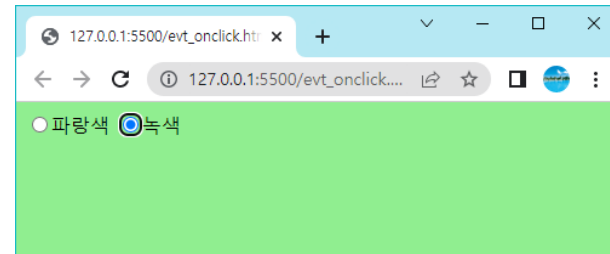
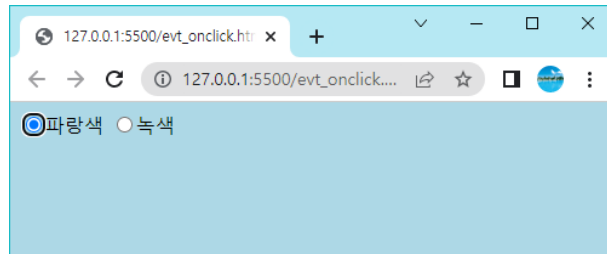
click 이벤트

- 사용자가 HTML 요소를 클릭하면 미리 정해놓은 코드가 실행되도록 할 수 있다.

→ 헤딩이 클릭되면 여기에 등록된 함수 change()가 호출된다.

```
<h1 onclick="change()">이것은 클릭 가능한 헤딩입니다.</h1>
```

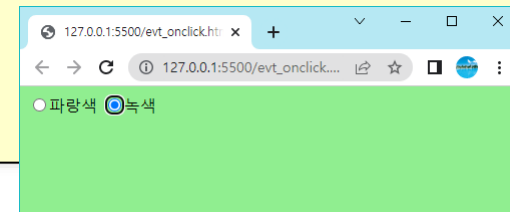
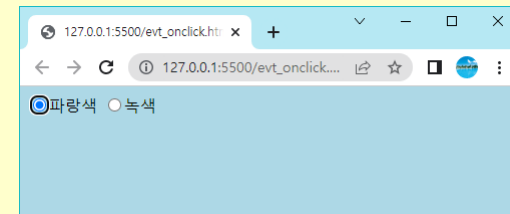
- 예제로 라디오 버튼 요소에 click 이벤트를 설정하여 보자.
사용자가 라디오 버튼을 클릭하면 <body> 요소의 색상이 변경되도록 한다.



예제

```
<!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>
```

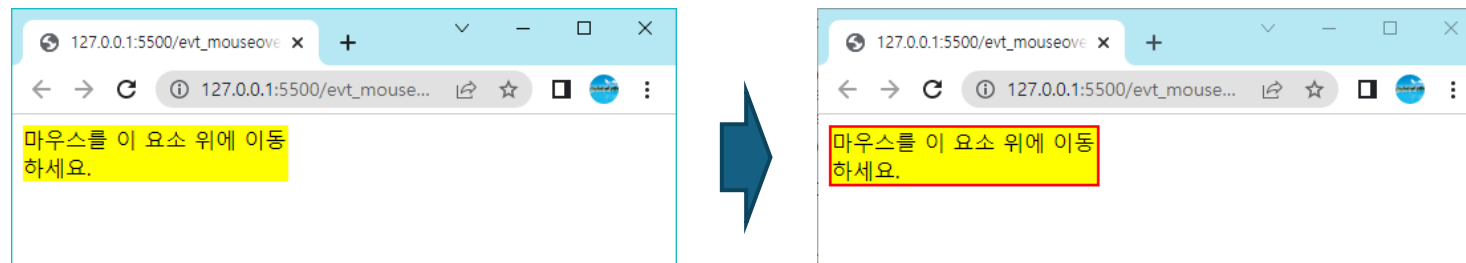


mouseover 이벤트

- mouseover와 mouseout 이벤트는 사용자가 HTML 요소 위에 마우스를 올리거나 요소를 떠날 때 발생.



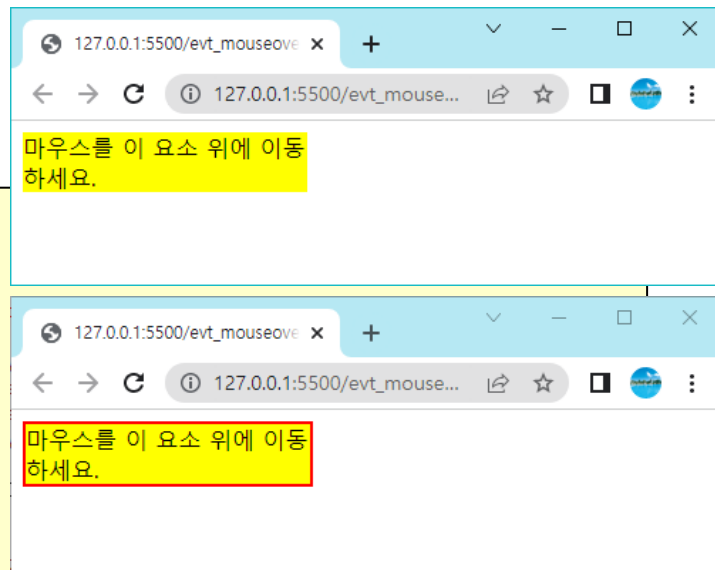
- 마우스 커서를 올리면 경계선이 표시되고 마우스 커서가 나가면 경계선이 사라지는 예제는 다음과 같이 작성한다.



예제

```
<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>
```



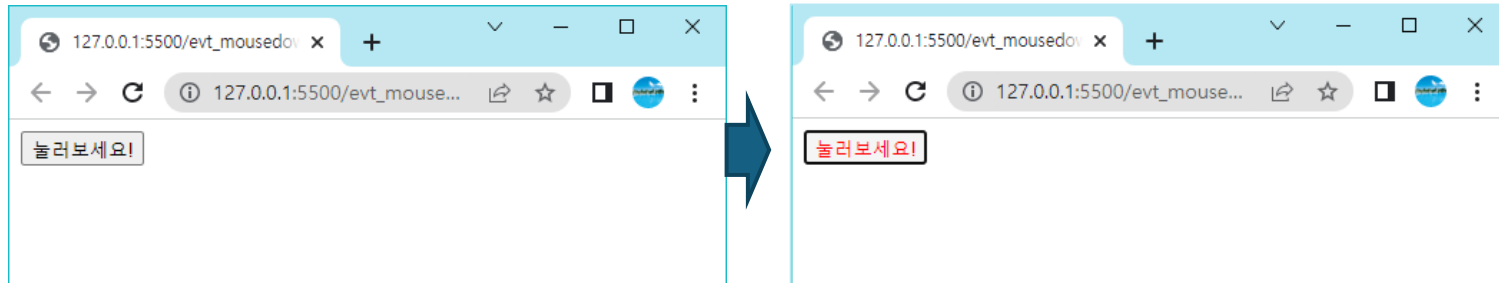
mousedown, mouseup 이벤트

- 먼저 마우스 버튼이 클릭되면 mousedown 이벤트가 발생
- 이어서 마우스 버튼이 떼어지면 mouseup 이벤트가 발생
- 마지막으로 마우스 클릭이 완료되면서 click 이벤트가 발생



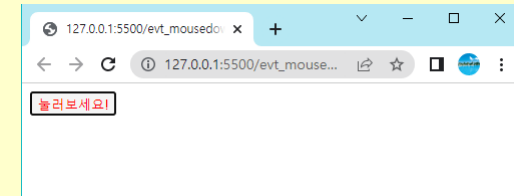
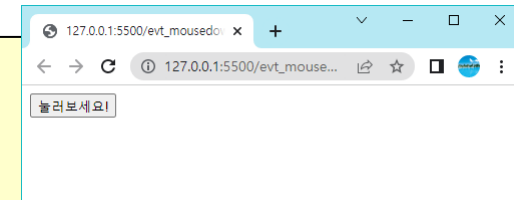
예제

- 마우스 커서를 올리면 경계선이 표시되고
마우스 커서가 나가면 경계선이 사라지는 예제는
다음과 같이 작성한다.



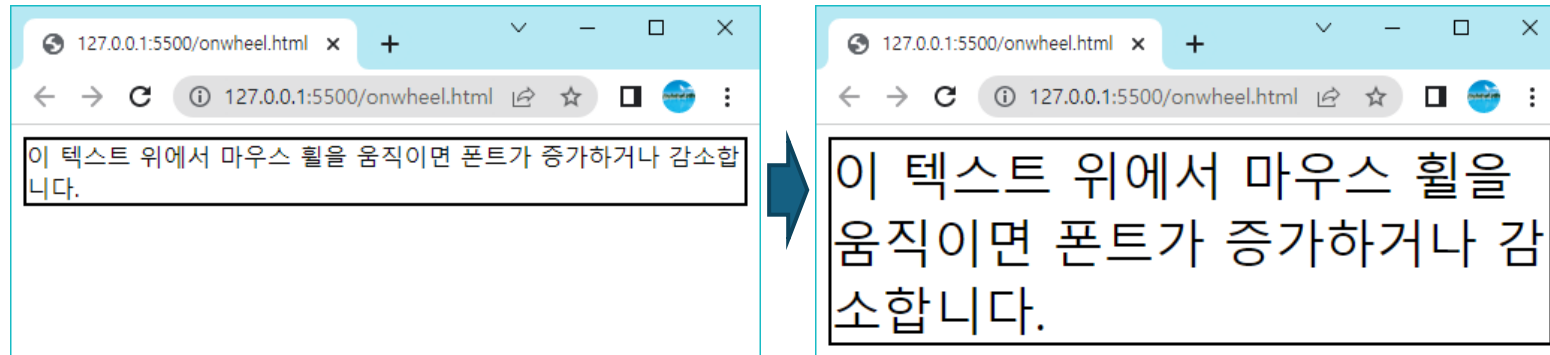
예제

```
<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>
```



wheel 이벤트

- 마우스의 휠을 돌리면 wheel 이벤트가 발생.
- <div> 요소 위에서 마우스 휠을 조작하면 폰트가 증가되는 예제를 작성해보자.

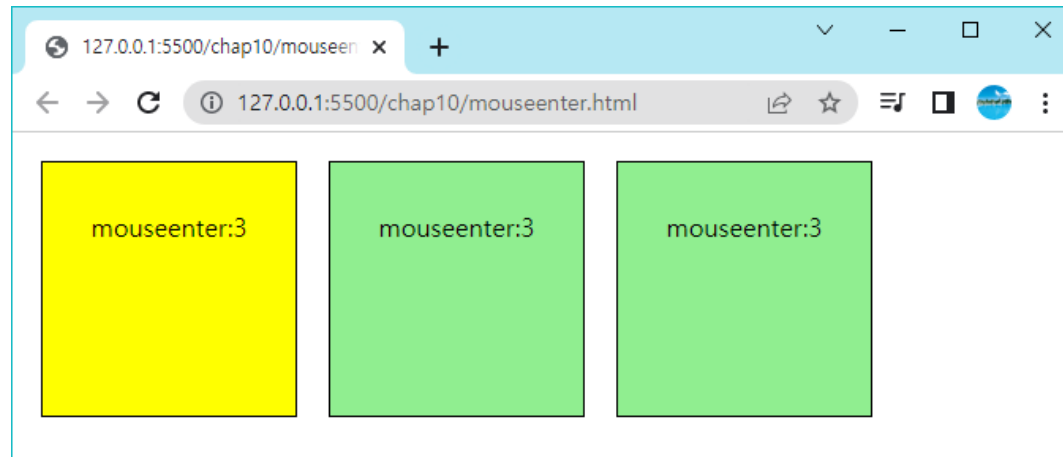


예제

```
<!DOCTYPE html>
<html>
<head>
  <style>
    #div1 {
      border: 2px solid black;
    }
  </style>
</head>
<body>
  <div id="div1">이 텍스트 위에서 마우스 휠을 움직이면 폰트가 증가하거나 감소합니다.</div>
  <script>
    document.getElementById("div1").addEventListener("wheel", sub);
    let size = 30;
    function sub(e) {
      if (e.wheelDelta > 0)
        size++;
      else
        size--;
      this.style.fontSize = size + "px";
    }
  </script>
</body>
</html>
```

mouseenter와 mouseleave

- mouseenter는 요소 위로 마우스 포인터를 이동할 때 발생한다.
- 3개의 박스를 만들고 mouseenter가 발생할 때마다, 배경색을 바꾸고 카운터를 1씩 증가시키는 코드를 작성해보자.



예제

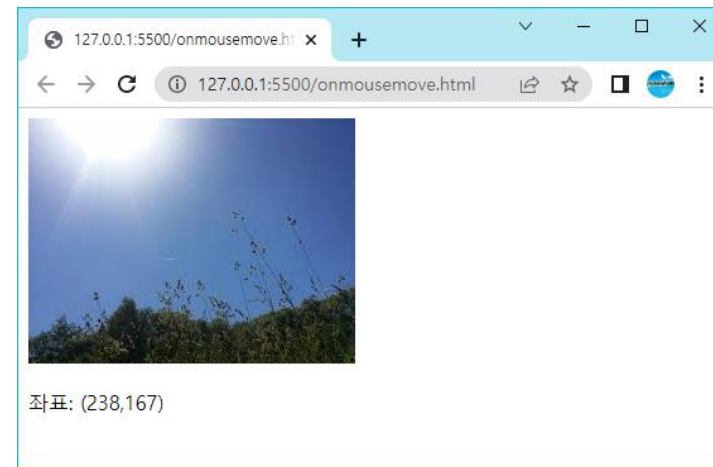
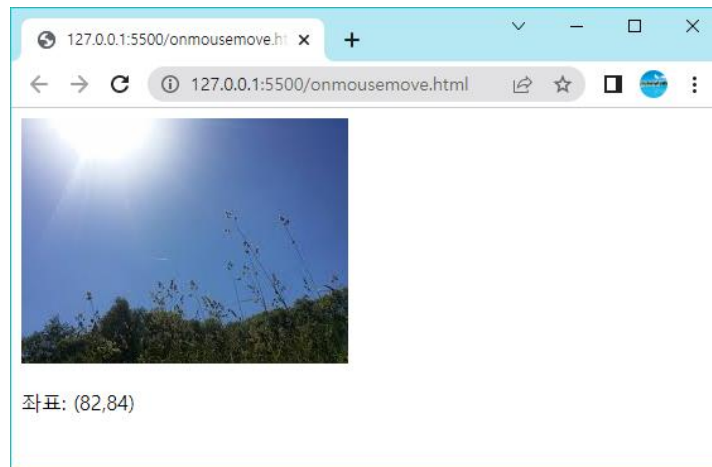
```
<!DOCTYPE html>
<html>
<head>
  <style>
    div {
      width: 100px;      height: 100px;      border: 1px solid black;
      margin: 10px;      float: left;
      padding: 30px;      text-align: center;
      background-color: lightgreen;
    }
  </style>
</head>
<body>
  <div onmouseenter="enter1(this)"onmouseleave="leave(this)"> </div>
  <div onmouseenter="enter2(this)"onmouseleave="leave(this)"> </div>
  <div onmouseenter="enter3(this)"onmouseleave="leave(this)"> </div>
  <script>
    let x = 0;    let y = 0;    let z = 0;
    function enter1(obj) {
      x++;
      obj.innerHTML = "mouseenter:" + x;
      obj.style.background = "yellow";
    }
  </script>
</body>
</html>
```

예제

```
function enter2(obj) {  
    y++;  
    obj.innerHTML = "mouseenter:" + y;  
    obj.style.background = "yellow";  
}  
function enter3(obj) {  
    z++;  
    obj.innerHTML = "mouseenter:" + z;  
    obj.style.background = "yellow";  
}  
function leave(obj) {  
    obj.style.background = "lightgreen";  
}  
</script>  
</body>  
</html>
```

onmousemove 이벤트

- onmousemove 이벤트는 마우스의 좌표를 우리에게 전달해준다. 마우스가 움직이면 연속하여 발생된다.



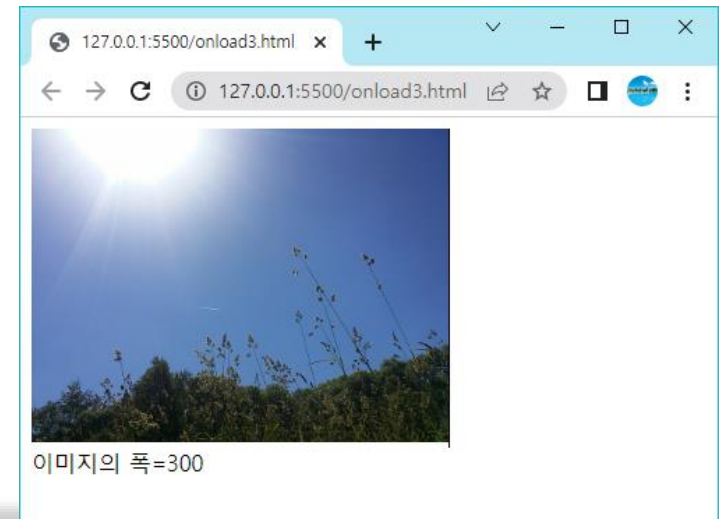
예제

```
<!DOCTYPE html>
<html>
<head>
  <style>
    #img1 {
      width: 200px;
      border: 1px solid black;
    }
  </style>
</head>
<body>
  <br>
  <p id="test"></p>
  <script>
    function sub(e) {
      let x = e.clientX;
      let y = e.clientY;
      let coor = "좌표: (" + x + ", " + y + ")";
      document.getElementById("test").innerHTML = coor;
    }
  </script>
</body>
</html>
```

load 이벤트

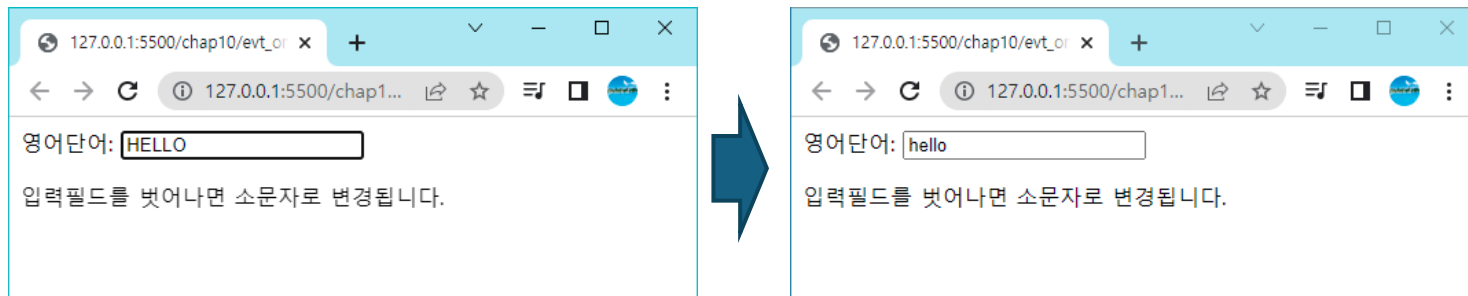
- HTML 문서가 로딩되거나, 사용자가 웹페이지를 떠나면 각각 load와 unload 이벤트가 발생.
- **load 이벤트는 문서의 로딩이 완료되면 발생하는 이벤트이다.**
- 자바스크립트 코드에서 아직 생성되지 않은 이미지나 요소를 미리 사용하면 안 되기 때문이다.

```
<!DOCTYPE html>
<html lang="en">
<body>
  
  <script>
    function loadImage() {
      document.getElementById('result').innerHTML
        = "이미지의 폭=" +
        document.getElementById('image').clientWidth;
    }
  </script>
  <div id="result"></div>
</body>
</html>
```



change 이벤트

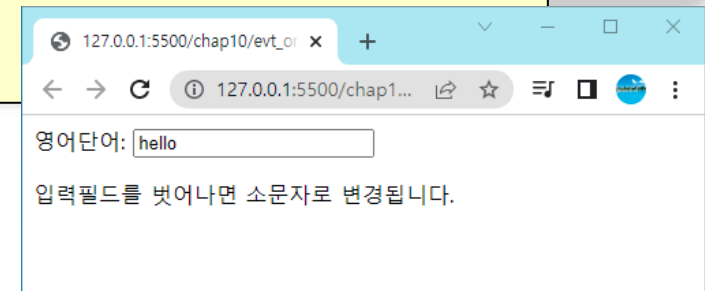
- change 이벤트는 입력 필드를 검증할 때 종종 사용된다.
- 아래 예제에서 사용자가 입력 필드의 내용을 변경하면 toLowerCase() 함수가 호출된다.



change 이벤트

```
<!DOCTYPE html>

<html>
<head>
<script>
  function sub()      {
    let 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>
```

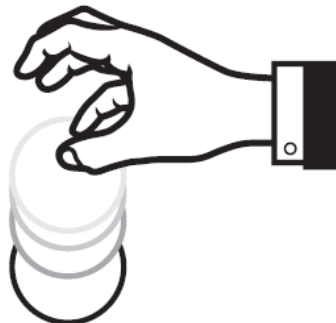


드래그와 드롭

- **드래그(drag)와 드롭(drop)** - 윈도우에서 아주 많이 사용하는 사용자 인터페이스 중의 하나
- 객체를 마우스로 끌어서 다른 애플리케이션에 놓는 것

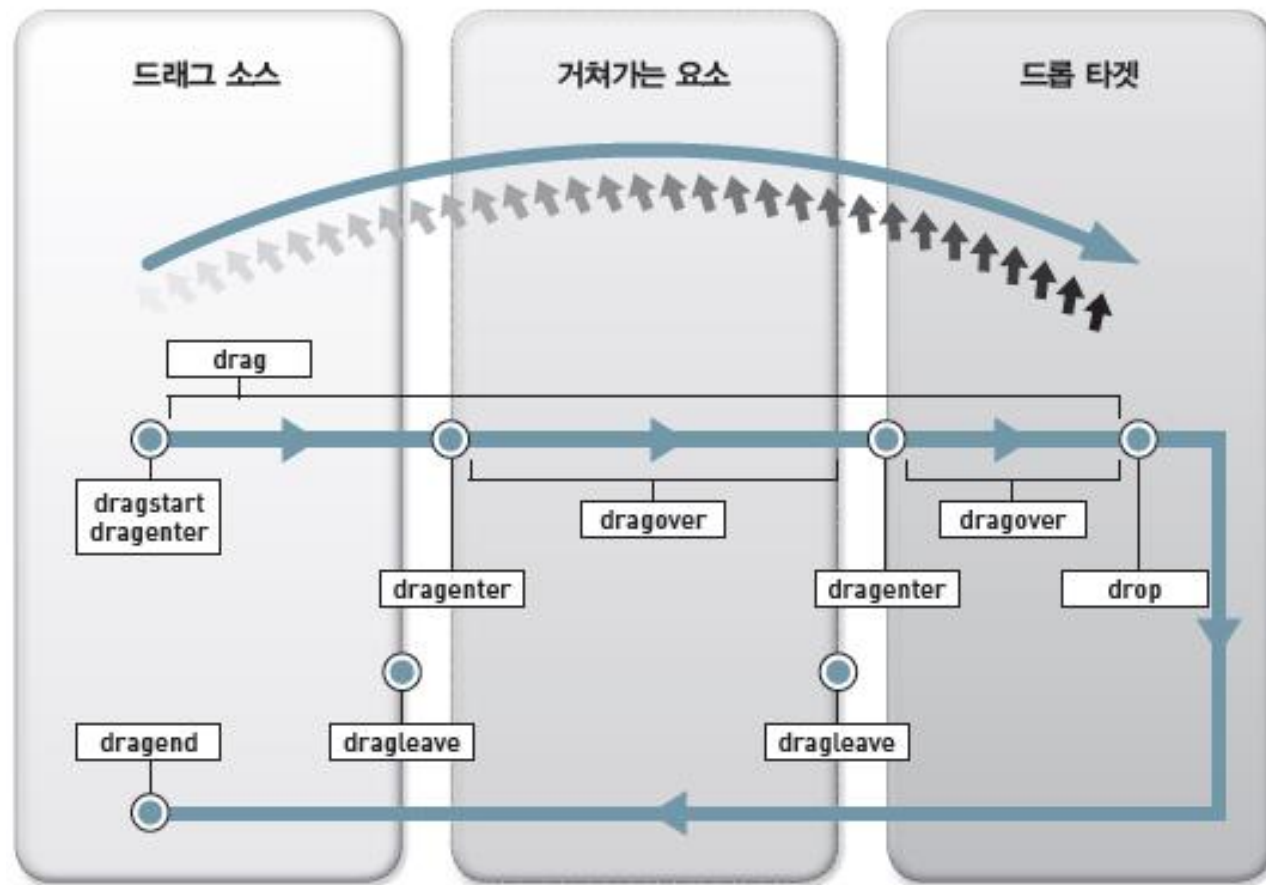


드래그



드롭

발생하는 이벤트



이벤트

- dragstart 이벤트
 - 사용자가 드래그를 시작할 때 발생. 이 때 해야 하는 가장 중요한 작업은 `dataTransfer` 객체에 `setData()` 호출을 통하여 데이터를 설정하는 작업.
- drag 이벤트
 - 드래그하는 도중에 계속해서 발생하는 이벤트.
- dragenter 이벤트
 - 마우스로 드래그하다가 새로운 요소 안으로 들어가면 `dragenter` 이벤트가 발생.
- dragover 이벤트
 - 드래그 동작 도중에 마우스가 다른 요소 위에 있다는 것을 의미.
- drop 이벤트
 - 반드시 처리해야 하는 이벤트. `drop` 이벤트는 사용자가 마우스 버튼을 놓았을 때 발생. `dataTransfer` 객체에서 `getData()` 메소드를 이용하여서 필요한 데이터를 꺼내면 됨

예제

- 웹 페이지 위에 상품을 드래그하여 쇼핑 카트에 넣을 수 있도록 웹 페이지를 작성하여 보자.



예제

```
<!DOCTYPE HTML>
<html>
<head>
  <style>
    #shopping_cart {
      width: 450px;
      height: 100px;
      padding: 10px;
      border: 1px dotted red;
    }
  </style>
  <script>
    function allowDrop(e) {
      e.preventDefault();
    }

    function handleDragStart(e) {
      e.dataTransfer.effectAllowed = 'move';
      e.dataTransfer.setData("Text", e.target.id);
    }
  </script>

```

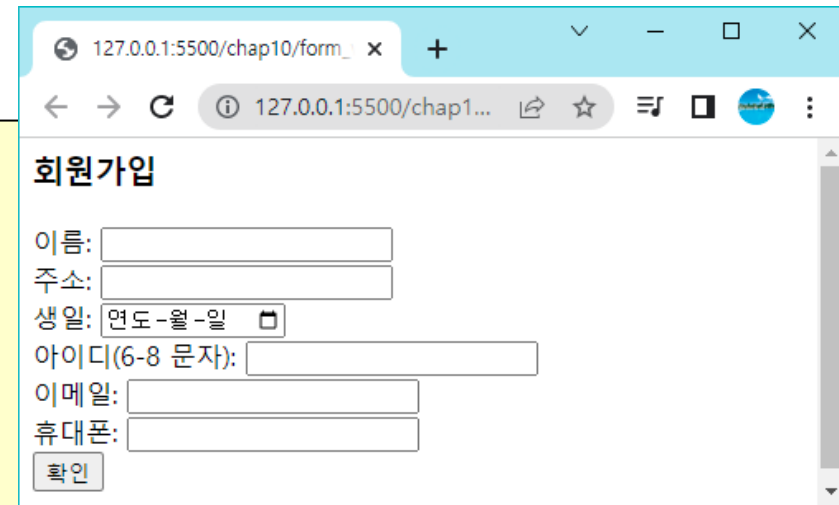
예제

```
function handleDrop(e) {  
    e.preventDefault();  
    let src = e.dataTransfer.getData("Text");  
    e.target.appendChild(document.getElementById(src));  
}  
</script>  
</head>  
<body>  
    <p>원하는 물건을 끌어서 옮기세요.</p>  
    <div id="shopping_cart"  
ondrop="handleDrop(event)" ondragover="allowDrop(event)"> </div>  
    <br>  
      
      
      
</body>  
</html>
```


입력값의 유효성 검증

- 자바스크립트는 사용한 입력한 데이터를 서버로 보내기 전에, 클라이언트 컴퓨터에서 사전 검증하는데 많이 사용된다.
- 다음과 같은 회원 가입 페이지에서 아이디를 입력할 때, 정해진 문자 수를 초과할 수도 있고, 이메일 주소를 잘못 입력할 수도 있다.

```
<h3>회원가입</h3>
<form>
이름: <input type='text' id='name' /><br />
주소: <input type='text' id='addr' /><br />
생일: <input type='date' id='birthday' /><br />
아이디(6-8 문자):
<input type='text' id='username' /><br />
이메일: <input type='email' id='email' /><br />
휴대폰: <input type='tel' id='phone' /><br />
<input type='submit' value='확인' /><br />
</form>
```



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5500/chap10/form_'. The page title is '회원가입' (Member Registration). The form contains the following fields and a button:

- 이름: (Name)
- 주소: (Address)
- 생일: (Birthdate) with a date picker icon
- 아이디(6-8 문자): (Username, 6-8 characters)
- 이메일: (Email)
- 휴대폰: (Mobile Phone)
- 확인 (Confirm) button

검증내용

- 사용자가 필수적인 필드를 채웠는가?
- 사용자가 유효한 길이의 텍스트를 입력하였는가?
- 사용자가 유효한 이메일 주소를 입력하였는가?
- 사용자가 유효한 날짜를 입력하였는가?
- 사용자가 숫자 필드에 텍스트를 입력하지 않았는가?

입력 양식 데이터 접근

- 사용자가 입력한 입력 양식 데이터에 접근하기 위해서는 입력 양식 안에 있는 필드의 id나 name 속성을 이용하여야 함

id 속성은 웹 페이지에서 요소들을 식별한다. → name 속성은 입력 양식 내부에서 필드들을 식별한다.

```
<input type="text" id="address" name="addr" />
```

주소:

```
<input type="text" name="addr" onclick="display(this.form)"/>
```

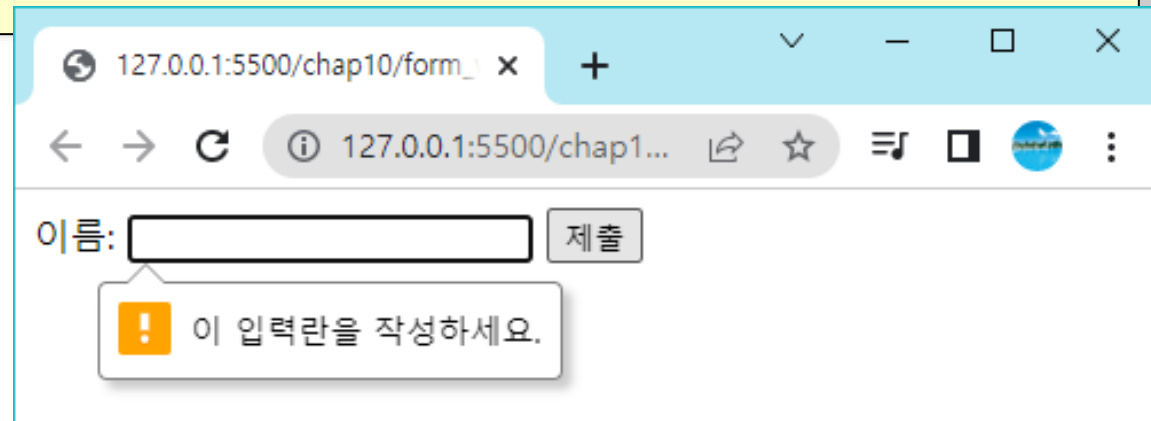
```
function display(form) {  
    alert(form["addr"].value);  
}
```

form 객체는 입력양식 안의 필드로 이루어진 배열로서 name을 식별자로 하여 필드값을 저장한다.

공백 검증

- 가장 기초적인 데이터 검증은 필드가 비어있는지를 체크하는 것

```
...  
<form>  
  이름: <input type='text' id='user' required>  
  <input type='submit' value='제출'>  
</form>
```



127.0.0.1:5500/chap10/form_ x +

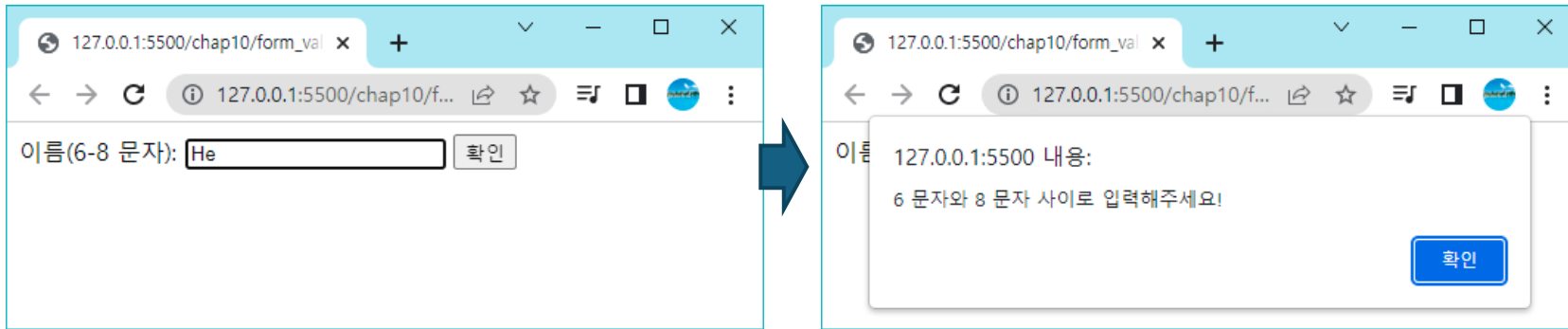
← → ↻ ⓘ 127.0.0.1:5500/chap1... ⌂ ☆ 🎵 📺 🌐 ⋮

이름: 제출

! 이 입력란을 작성하세요.

데이터 길이 검증

- 사용자가 정해진 개수의 문자만을 입력하도록 하는 경우도 많다. 예를 들어서 주민등록번호는 13자리이다.

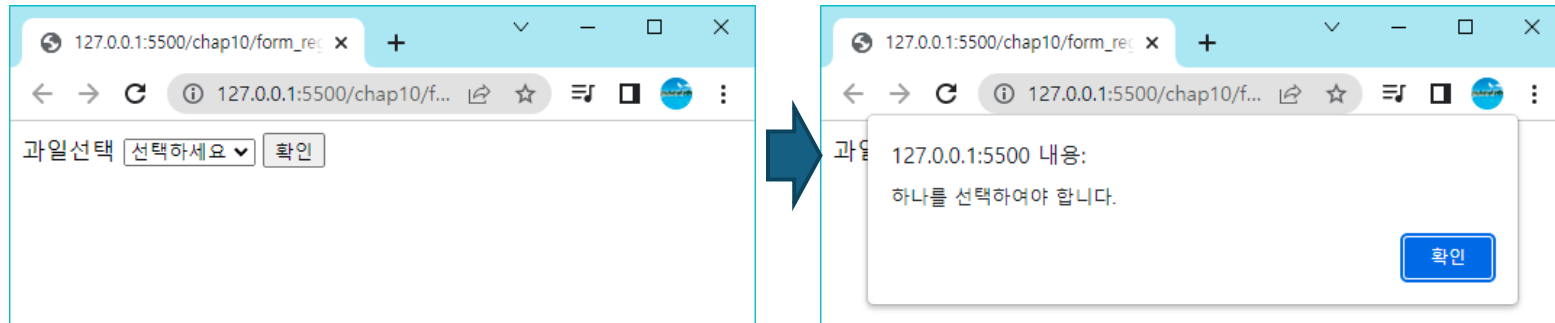


데이터 길이 검증

```
<script>
function checkLength(elem, min, max) {
    let s = elem.value;
    if (s.length >= min && s.length <= max) {
        return true;
    } else {
        alert(min + " 문자와 " + max + " 문자 사이로 입력해주세요!");
        elem.focus();
        return false;
    }
}
</script>
<form>
    이름(6-8 문자): <input type='text' id='name' />
    <input type='button'
        onclick="checkLength(document.getElementById('name'), 6, 8)"
        value='확인' />
</form>
```

선택 검증

- HTML select 요소에서 사용자가 선택을 했는지를 검증하려면 약간의 트릭이 필요하다.
- **첫 번째 옵션을 도움말로 두는 것이다.**

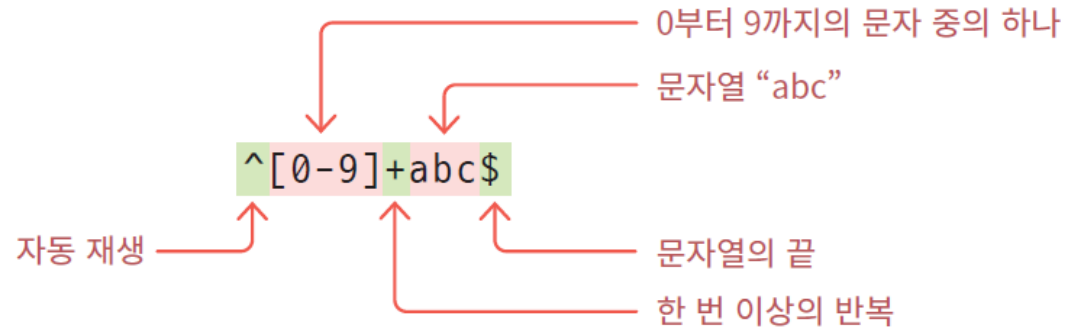


선택 검증

```
<script>
  function checkSelection(elem, msg) {
    if (elem.value == 0) {
      alert(msg);
      elem.focus();
      return false;
    } else {
      return true;
    }
  }
</script>
<form>
과일선택 <select id="fruits" class="required">
  <option value="0">선택하세요</option>
  <option value="1">사과</option>
  <option value="2">배</option>
  <option value="3">바나나</option>
</select>
<input type='button'
        onclick="checkSelection(document.getElementById('fruits'), '하나를 선택하여야 합니다.')"
        value='확인' />
</form>
```


정규식

- 정규식(regular expression)이란?
 - 특정한 규칙을 가지고 있는 문자열들을 표현하는 수식.



정규식

식	기능	설명
^	시작	문자열의 시작을 표시
\$	끝	문자열의 끝을 표시
.	문자	한 개의 문자와 일치
\d	숫자	한 개의 숫자와 일치
\w	문자와 숫자	한 개의 문자나 숫자와 일치
\s	공백문자	공백, 탭, 줄바꿈, 캐리지리턴 문자와 일치
[]	문자 종류, 문자 범위	[abc]는 a 또는 b 또는 c를 나타낸다. [a-z]는 a부터 z까지 중의 하나, [1-9]는 1부터 9까지 중의 하나를 나타낸다.

수량 한정자	기능	설명
*	0회 이상 반복	"a*"는 "", "a", "aa", "aaa"를 나타낸다.
+	1회 이상	"a+"는 "a", "aa", "aaa"를 나타낸다.
?	0 또는 1회	"a?"는 "", "a"를 나타낸다.
{m}	m회	"a{3}"는 "aaa"만 나타낸다.
{m, n}	m회 이상 n회 이하	"a{1, 3}"는 "a", "aa", "aaa"를 나타낸다.
(ab)	그룹화	(ab)*은 "", "ab", "abab" 등을 나타낸다.

정규식의 예

- 만약 계좌번호를 검증한다고 가정하자(계좌번호는 10자리의 숫자로 되어 있다고 가정하자).
- 이것은 정규식 `/^\d{10}$/`으로 나타낼 수 있다.

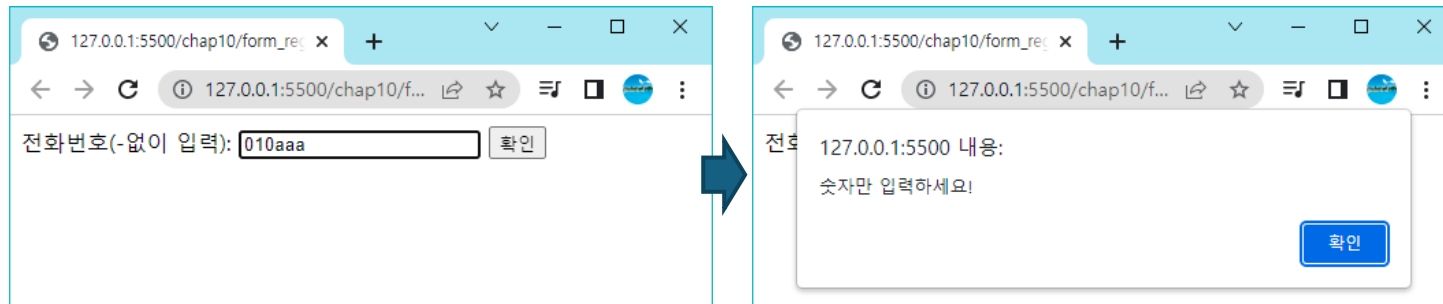
```
let exp = /^\d{10}$/;
if( !exp.test(field.value) ){
    alert("오류!!");
    return false;
}
```

→ 이 리터럴은 RegExp 객체를 생성한다.

→ RegExp 객체의 test() 메소드는 일치하면 true를 반환한다.

숫자 여부 검사하기

- 숫자로만 된 데이터를 검증하는 가장 좋은 방법은 정규 표현식을 사용하는 것.
- `/^[0-9]+$`은 문자열이 모두 숫자로만 되어 있어야 일치함.



예제

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

HTML5 위치 정보

- **위치정보(Geolocation)**은 자신의 위치를 웹 사이트와 공유
- 현재 지역의 날씨, 유명한 맛집 등의 정보를 제공받을 수 있다.



geolocation 객체

- `var geolocation = navigator.geolocation;`

메소드	설명
<code>getCurrentPosition()</code>	사용자의 현재 위치 정보를 반환한다.
<code>watchPosition()</code>	장치의 현재 위치에 대한 정보를 주기적으로 반환한다.
<code>clearWatch()</code>	현재 진행 중인 <code>watchPosition()</code> 실행을 중지한다.

getCurrentPosition() 메소드

- 위치 정보를 가져올 때 사용하는 메소드
- 위치 정보를 처리하는 함수를 인수로 전달.
 - 예를 들어서 다음과 같이 사용할 수 있다.

```
function getLocation() {  
  let geolocation = navigator.geolocation;  
  geolocation.getCurrentPosition(showLocation, errorHandler);  
}
```

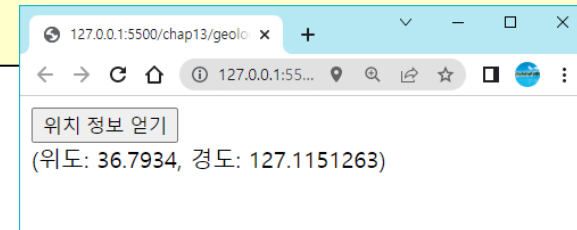
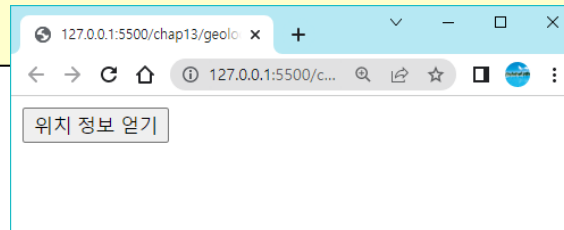
```
function showLocation( position ) {  
  let latitude = position.coords.latitude;  
  let longitude = position.coords.longitude;  
  ...  
}
```


position 인수

속성	타입	설명
<code>coords</code>	<code>objects</code>	장치의 위치정보를 가지고 있는 객체이다.
<code>coords.latitude</code>	<code>Number</code>	위도 정보이다. 단위는 도(decimal degree)이다. 범위는 $[-90.00, +90.00]$.
<code>coords.longitude</code>	<code>Number</code>	경도 정보이다. 단위는 도(decimal degree)이다. 범위 $[-180.00, +180.00]$.
<code>coords.altitude</code>	<code>Number</code>	고도 정보이다. 단위는 미터이다.
<code>coords.heading</code>	<code>Number</code>	북쪽으로부터 시계방향으로 방위각

예제

```
<!DOCTYPE html>
<html>
<body>
  <button onclick="getGeolocation()">위치 정보 얻기</button>
  <div id="target"></div>
  <script>
    let myDiv = document.getElementById("target");
    function getGeolocation() {
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showLocation);
      }
    }
    function showLocation(location) {
      myDiv.innerHTML = "(위도: " + location.coords.latitude +
        ", 경도: " + location.coords.longitude + ")"
    }
  </script>
</body>
</html>
```

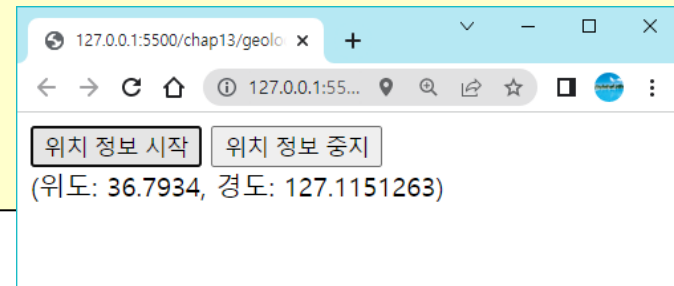


이동하면서 위치 정보를 얻는 방법

- 사용자가 이동하면서 주기적으로 위치정보를 얻으려면 geolocation 객체의 watchPosition()을 호출하여 콜백 메소드를 등록.
- 콜백 메소드가 주기적으로 호출되면서 업데이트된 위치정보를 얻음
- watchPosition() - 사용자의 현재 위치를 연속하여 출력.
자동차처럼 사용자가 이동하고 있으면 계속 업데이트된 위치를 반환.
- clearWatch() - watchPosition() 메소드를 중지함.

예제

```
<!DOCTYPE html>
<html>
<body>
  <button onclick="startGeolocation()">위치 정보 시작 </button>
  <button onclick="stopGeolocation()">위치 정보 중지 </button>
  <div id="target"> </div>
  <script>
    let id;
    let myDiv = document.getElementById("target");
    function startGeolocation() {
      if (navigator.geolocation) {
        id = navigator.geolocation.watchPosition(showGeolocation);
      }
    }
    function showGeolocation(location) {
      myDiv.innerHTML = "(위도: " + location.coords.latitude +
        ", 경도: " + location.coords.longitude + ")";
    }
    function stopGeolocation() {
      if (navigator.geolocation) {
        navigator.geolocation.clearWatch(id);
      }
    }
  </script>
</body>
</html>
```



HTML5 웹 워커

- 가끔은 자바스크립트에서 시간이 많이 걸리는 작업들을 할 때가 있음
- 이런 작업들을 자바스크립트에서 한다면 웹 페이지는 자바 스크립트가 완료 될 때까지 응답 하지 않게 됨
- **웹 워커(web worker):** 자바스크립트에 백그라운드에서 실행되는 스레드(thread)를 도입한 것



소수 구하기 예제

worker.js

```
// 소수를 찾는 자바스크립트 소스
let n = 1;
search: while (true) {
  n += 1;
  for (let i = 2; i <= Math.sqrt(n) ; i += 1)
    if (n % i == 0)
      continue search;
  // 소수를 발견할 때마다 바로 웹페이지로 전달한다.
  postMessage(n);
}
```

소수 구하기 예제

webworker.html

```
<!DOCTYPE HTML>
<html>
<head>
  <title>웹워커 예제 </title>
</head>
<body>
  <button onclick="startWorker()">웹워커 시작 </button>
  <button onclick="stopWorker()">웹워커 종료 </button>
  <p>현재까지 발견된 가장 큰 소수는
    <output id="result"> </output>
  </p>
```

소수 구하기 예제

webworker.html

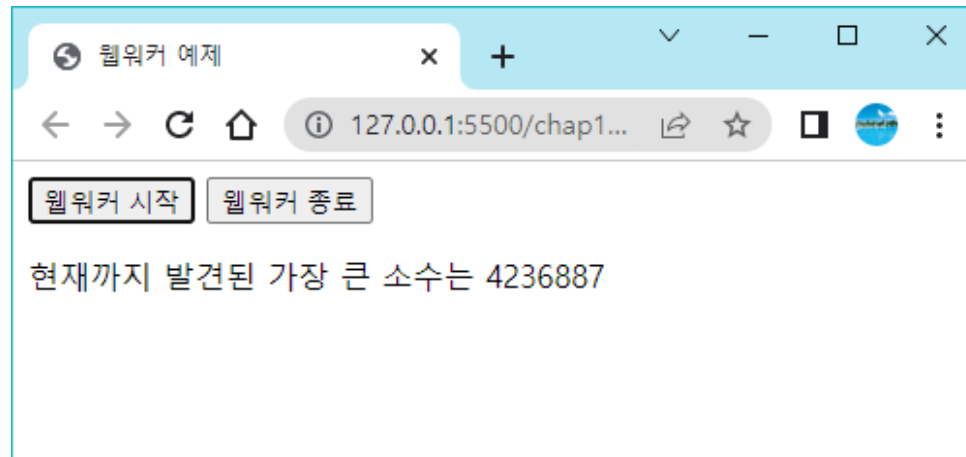
```
<script>
  let w;

  function startWorker() {
    if (typeof (Worker) !== "undefined") {
      if (typeof (w) === "undefined") {
        w = new Worker("worker.js");
      }
      w.onmessage = function (event) {
        document.getElementById("result").innerHTML = event.data;
      };
    }
    else {
      document.getElementById("result").innerHTML = "웹브라우저가 웹워커를 지원하지 않음";
    }
  }

  function stopWorker() {
    w.terminate();
  }
</script>
</body>
</html>
```


소수 구하기 예제

- 이번 예제는 반드시 웹서버를 통하여 실행하여야 함.
즉 라이브 서버를 통하여 실행하여야 함.
단순히 더블 클릭만 하면 웹워커가 실행되지 않음.



웹워커의 용도

- 영상 처리(image processing)
 - 캔버스나 비디오 요소에서 추출된 데이터를 사용하여 어떤 영상 처리를 수행할 수 있다. 이때는 이미지를 여러 개의 작은 조각으로 나누어서 각 조각들을 웹워커에게 줄 수도 있다. 멀티-코어 CPU라면 물리적으로도 웹워커들이 동시에 수행될 수 있다.
- 대용량 데이터 처리
 - XMLHttpRequest 호출 후에 파싱해야 할 대용량 데이터가 존재하는 경우에 사용할 수 있다.
- 텍스트 분석
 - 사용자가 입력하는 즉시 텍스트를 사전에서 검색한다. 자동적인 오류 수정 등이 가능하다.
- 데이터베이스 요청 동시 수행
 - 로컬 데이터베이스에 대한 요청들을 동시에 수행한다.

이번 장에서 배운 것

- HTML 요소를 클릭하였을 때 발생하는 이벤트를 처리할 수 있나요?
 - click 이벤트를 처리하면 된다.
 - 이벤트 리스너는 인라인으로도 작성이 가능하고 무명 함수로도 만들 수 있다.
- 마우스를 요소 안으로 움직였을 때 발생하는 이벤트를 처리할 수 있나요?
 - mouseenter나 mousehover 이벤트를 처리하면 가능하다.
- 사용자가 8글자 이상을 입력하였는지 검사할 수 있나요?
 - 문자열의 길이를 계산하여서 8글자 이상인지를 체크한다.
- 사용자가 숫자만 입력하였는지 검사할 수 있나요?
 - 문자열의 글자들을 하나씩 검사하여도 되고, 복잡한 검증은 정규식을 사용하는 것이 편리하다.

Q & A