이벤트

- 자바스크립트 이벤트 처리는 웹 개발자에게 가장 까다로운 부분이었다
  - 브라우저마다 이벤트 처리 방식이 다르기 때문에
  - 이벤트 객체를 전달하는 방식뿐만 아니라 이벤트 객체의 속성 정보도 달랐음
- jQuery 라이브러리는 자체적으로 추상화된 jQuery 이벤트 객체 제공
  - 개발자가 브라우저 호환성을 일일이 신경 쓰지 않아도 됨.
  - 단, jQuery 이벤트 처리 방식을 사용해야만 한다.

자바스크립트 이벤트 모델(1)

자바스크립트 이벤트 처리방식

- 인라인 이벤트 : HTML 마크업(Markup)에 함수 호출 코드를 직접 작성하는 방식
- 인라인 이벤트의 몇가지 문제점
  - 비간섭적 자바스크립트 원칙 위배
    - 구조와 행동을 분리하라는 것이다.
    - 구조는 HTML 마크업이고, 행동은 실행되는 자바스크립트 코드이다.
  - 크로스 브라우저 환경 지원이 힘들다.
    - 브라우저마다 이벤트 객체를 전달하는 방식과 이벤트 객체의 속성이 다르다.
    - 이것을 개발자가 조건 처리하여 직접 해결해야 한다.
  - this, arguments와 같이 함수가 실행될 때 바인딩되는 값의 문제이다
    - 대부분의 개발자는 this로 바인딩되는 객체가 이벤트가 발생한 요소(Element)이기를 바란다
    - 인라인 이벤트 처리 함수에서의 this는 전역(Global) 객체임.
  - 함수 이름의 충돌 문제
    - 대부분 개발자들은 인라인 이벤트 처리 함수를 대부분 전역에 작성함.
    - 이름 충돌 가능성이 높아짐.

자바스크립트 이벤트 모델(2)

- 예제 11-01

```
07: function inline_event() {
08:   console.log(this);
09: }
```

```
13:   <input type="button" value="인라인 이벤트" onclick="inline_event()" /> <br />
```

  - this가 HTML 요소가 되기를 원하겠지만 this는 전역 객체!!

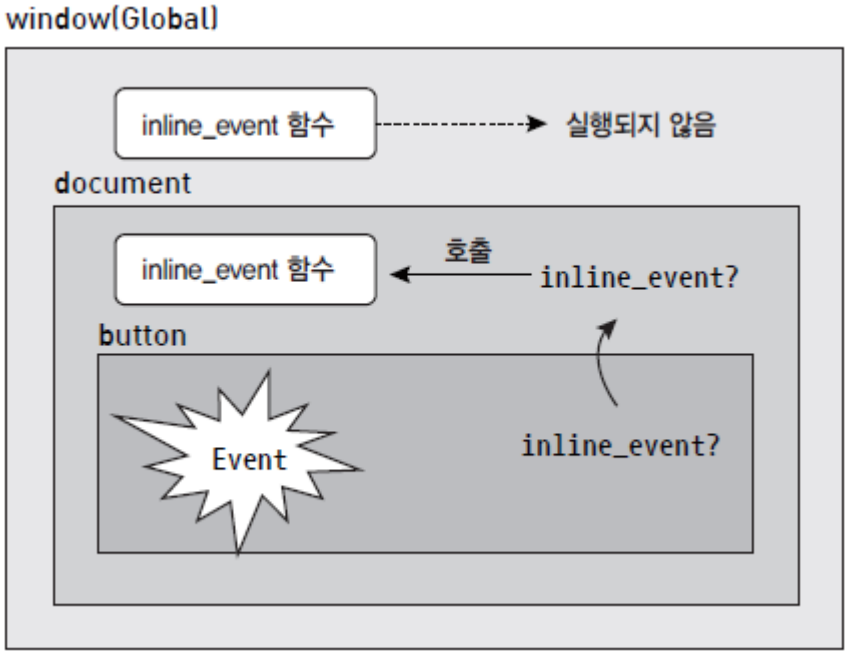
- 예제 11-02

```
07: function inline_event() {
08:   console.log(this);
09: }
10:
11: document.inline_event = function() {
12:   console.log("Document");
13: };
```

  - document.inline_event 실행!!

자바스크립트 이벤트 모델(3)

- 예제 11-02 실행 결과 이해



- 그렇다면 document.inline_event 메서드를 안만들면 될일 아닌가?
- 모든 메서드의 충돌을 피할 자신이 있다면 그렇게 해도 된다.
  - 대규모 웹앱을 개발하게 되면 쉽지 않은 일이다.

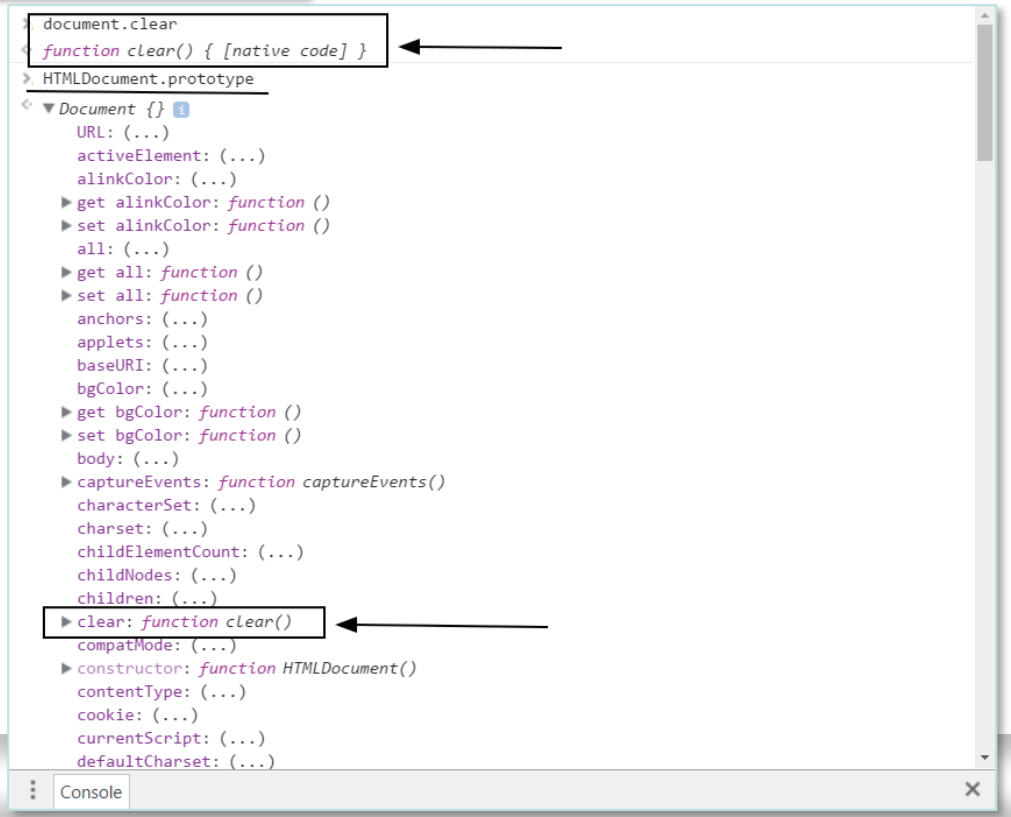
자바스크립트 이벤트 모델(4)

- 예제 11-03 : 이름 충돌 두번째

```
07: function clear() {
08:    console.log("Clear!!");
09: }
```

```
13:    <input type="button" value="인라인 이벤트" onclick="clear()" /> <br />
```

- 이 코드는 정상 실행되지 않음. document 객체에 이미 clear() 메서드가 존재하기 때문이다.
- document 객체의 prototype을 거슬러 올라가보면 clear() 메서드가 존재한다.
- 즉 prototype 모두에 존재하는 메서드 이름의 중복을 피해야 한다.



자바스크립트 이벤트 모델(5)

고전 이벤트 모델

```
14:   a.onclick = function(e) {
15:      document.write("고전!!");
16:   };
```

표준 이벤트 모델

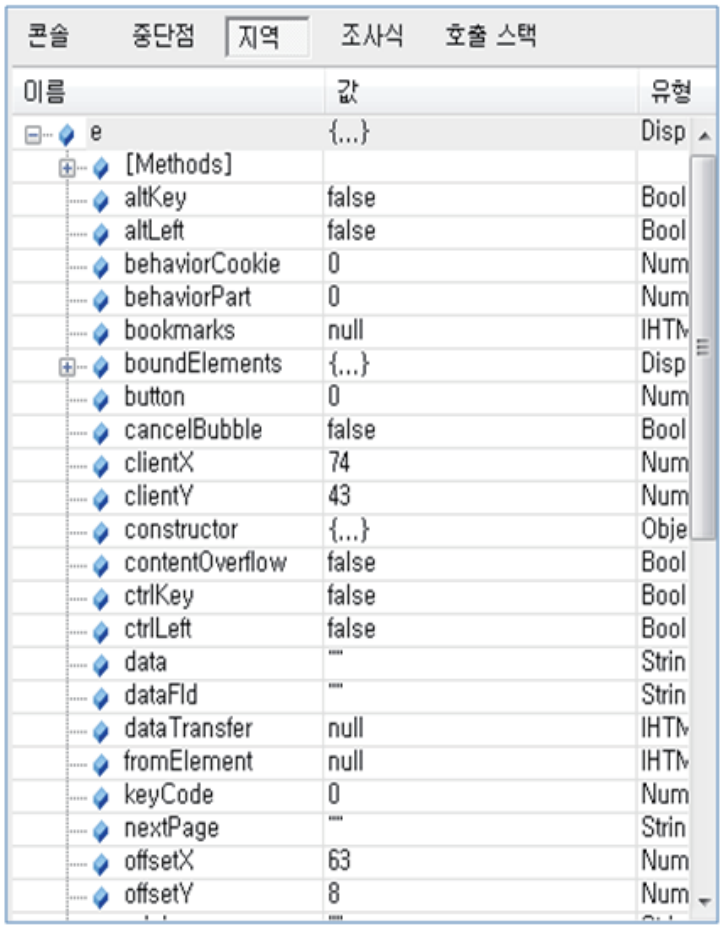
```
18:   if (b.addEventListener) {
19:      b.addEventListener("click", function(e) {
20:         document.write("표준 addEventListener!!");
21:      });
22:   } else {
23:      b.attachEvent("onclick", function(e) {
24:         document.write("표준 attachEvent!!");
25:      });
26:   }
```

- 두 가지 방법 모두 크로스 브라우저 처리에 문제가 있다.
  - ~IE8 : attachEvent 메서드, NonIE : addEventListener 메서드
  - 이벤트 객체의 정보도 브라우저마다 조금씩다르다.(특히 IE)

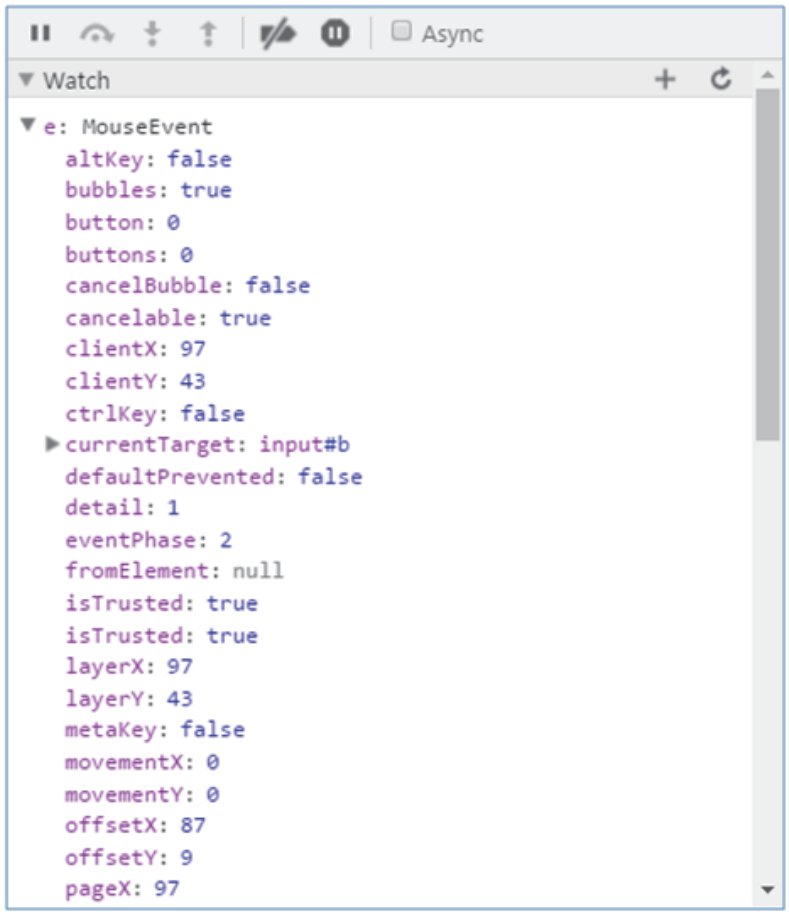
자바스크립트 이벤트 모델(6)

- 브라우저의 이벤트 객체 정보의 차이

IE 8



Chrome 49



자바스크립트 이벤트 모델(7)

jQuery 이벤트 처리

- 앞에서 기술한 4가지 문제점 모두 해결
- 크로스 브라우저 지원
  - jQuery 이벤트 처리코드는 어느 브라우저에서나 작 작동함.
  - jQuery 이벤트 객체는 각 브라우저의 서로 다른 이벤트 객체를 추상화하여 제공

브라우저 점유율

이 책에서는 IE(Internet Explorer)의 가장 낮은 버전을 IE 8로 설정했다. IE 6,7은 고려하지 않는다. 2016년 현재 (2016년 2월 기준) 우리나라의 데스크톱 브라우저 점유율을 살펴보면 다음과 같다

http://gs.statcounter.com/#browser_version-KR-monthly-201602-201602-bar

IE 11 : 28.8%

Chrome 48 : 28.28%

IE 10 : 14.77%

IE 9 : 9.92%

IE 8 : 4.83%

jQuery 이벤트 설정(1)

단축 이벤트와 이벤트 연결 메서드

- click, keyup 등 이미 사용해본 적 있음
  - 공식적인 이벤트 연결 메서드 : bind(), on()
  - 단축 이벤트 메서드 : click(), keyup()
  - 다음 두 코드는 동일한 기능

```
$("#inquery").click(function() { })
$("#inquery").on("click", function() { })
```

[**표 11-01** : 단축 이벤트 메서드 목록]

종류	이벤트
브라우저, 문서 로딩	resize( ), scroll( ), ready( )
입력 폼	submit( ), blur( ), change( ), focus( ), select( )
키보드	keydown( ), keyup( ), keypress( )
마우스	click( ), dblclick( ) mousedown( ), mouseup( ) mouseenter( ), mouseleave( ), mouseover( ), mouseout( ) hover( ), focusin( ), focusout( )

* jQuery 최신 버전에서 기능 지원 중단이 예고된(deprecated) 이벤트는 제외했다.

jQuery 이벤트 설정(1)

- 단축 이벤트 메서드에 터치가 보이지 않음. 모바일 브라우저 미지원?
  - 단축 이벤트 메서드는 자주 사용하는 이벤트를 좀 더 간단하게 사용하기 위해서 제공됨.
  - 내부적으로 단축 메서드는 모두 on() 메서드를 호출함.

```
> $.prototype.click
< function ( data, fn ) {
        return arguments.length > 0 ?
            this.on( name, null, data, fn ) :
            this.trigger( name );
    }
```

- 예제 11-05 : 예제 07-11을 복사하여 단한줄 수정

```
$("#inquery").click(function() {  });
→ $("#inquery").on("click", function() {  });
```

```
$("#inquery").bind("click", function() {  });
```

jQuery 이벤트 설정(2)

bind(), on() 메서드의 장점

- 다중 이벤트 설정
  - 한번에 여러 개의 이벤트를 설정할 수 있음

```
17:   $("#a").find("button.default").on("mouseenter  mouseleave", function() {
18:      $(this).toggleClass("hover");
19:   });
```

```
01: $("#a").find("button.default").on({
02:   mouseenter : function() {
03:      $(this).addClass("hover");
04:   },
05:   mouseleave : function() {
06:      $(this).removeClass("hover");
07:   }
08: });
```

```
01: $("#a").find("button.default").on("mouseenter", function() {
02:   $(this).addClass("hover");
03: }).on("mouseleave", function() {
04:   $(this).removeClass("hover");
05: });
```

jQuery 이벤트 설정(3)

이벤트 연결 해제 메서드

- off(), unbind() 메서드 : 이벤트 연결을 해제한다. 사용방법 동일

```
off()
off(eventtype)
off(eventtype, function)
```

- one()
  - 이벤트 연결 설정을 한 번만 수행하는 메서드
  - 이벤트가 발생해서 함수가 호출되고 나면 자동으로 이벤트 연결 설정이 해제

```
one(eventtype, function)
```

- 예제 11-17

```
$("#test1").one("click", function(e) {
   console.log(e);
});
```

==

```
$("#test1").on("click", function(e) {
   console.log(e);
   $("#test1").off("click");
});
```

jQuery 이벤트 설정(4)

이벤트 강제 실행

```
trigger(eventtype)

trigger(eventtype, parameter)
```

- 마우스를 클릭하지 않고도 click 이벤트에 연결된 함수를 호출할 수 있음
- 이벤트 객체가 제공하는 정보는 제한적임.
- 예제 11-07을 실행한 후 콘솔에서 직접 trigger()를 호출해 봄

버튼을 클릭했을 때

```
11-07.html:19
j…y.Event {originalEvent: MouseEvent, type: "click", timeStamp: 5575.225,
jQuery111106013182738838214: true, toElement: button#test1…}
  altKey: false
  bubbles: true
  button: 0
  buttons: 0
  cancelable: true
  clientX: 159
  clientY: 79
  ctrlKey: false
▶ currentTarget: button#test1
  data: undefined
▶ delegateTarget: button#test1
  eventPhase: 2
  fromElement: null
▶ handleObj: Object
▶ isDefaultPrevented: function returnFalse()
  jQuery111106013182738838214: true
  metaKey: false
```

trigger("click") 했을 때

```
> $("#test1").trigger("click")

j…y.Event {type: "click", timeStamp: 1458371081368,
jQuery111100540385154526337 2: true, isTrigger: 3, namespace: ""…}
▶ currentTarget: button#test1
  data: undefined
▶ delegateTarget: button#test1
▶ handleObj: Object
  isTrigger: 3
  jQuery111100540385154526337 2: true
  namespace: ""
  namespace_re: null
  result: undefined
▶ target: button#test1
  timeStamp: 1458371081368
  type: "click"
▶ __proto__: Object
```

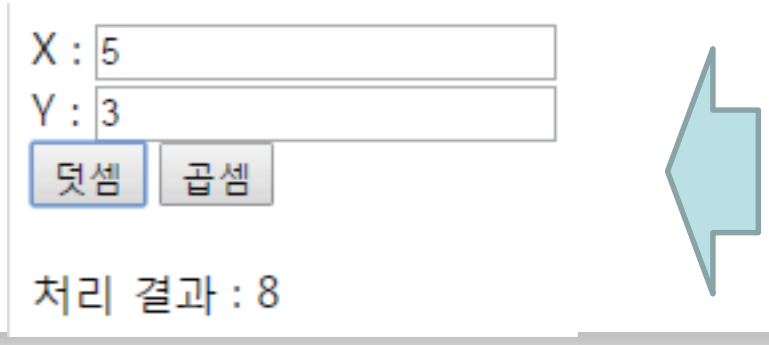
jQuery 이벤트 설정(5)

이벤트 데이터 전달

```
click(eventData, handlerFunction)

on(eventtype, selector, eventData, handlerFunction)
```

- eventData 파라미터
  - 대부분의 값을 전달할 수 있음
  - 전달한 이벤트 데이터는 호출되는 함수의 이벤트 객체를 통해 data 속성값으로 전달됨



예제 11-08

```
09:  function calcHandler(e) {
10:     var x = parseInt($("#x").val());
11:     var y = parseInt($("#y").val());
12:     var result = 0;
13:
14:     if (e.data == "+") {
15:        result = x+y;
16:     } else if (e.data == "*") {
17:        result = x*y;
18:     } else {
19:        throw new Error("처리 불가능한 연산!!");
20:     }
21:     $("#result").html(result);
22:  }
24:  $("#add").click("+", calcHandler);
25:  $("#multiply").click("*", calcHandler);

30:  X : <input type="text" id="x" value="5" /><br />
31:  Y : <input type="text" id="y" value="3" /><br />
32:  <button id="add">덧셈</button>
33:  <button id="multiply">곱셈</button><br /><br />
34:  처리 결과 : <span id="result">0</span>
```

jQuery 이벤트 설정(6)

- 이벤트 데이터 메서드를 on()으로…

```
24:   $("#add").click("+", calcHandler);
25:   $("#multiply").click("*", calcHandler);
```

```
$("#add").on("click", undefined, "+", calcHandler);
$("#multiply").on("click", undefined, "*", calcHandler);
```

- 두번째 파라미터는 undefined나 null로 설정
- 두번째 파라미터는 이벤트 위임 처리할 대상을 지정할 때 사용함

jQuery 이벤트 설정(7)

이벤트 네임스페이스

- 이벤트 타입뿐만 아니라 추가로 식별력있는 이름을(namespace)를 부여할 수 있음
- 네임스페이스 이용 형태
  - 하나의 요소에 여러 개의 이벤트 처리 함수를 연결하면서 식별력을 가지도록 할 때
  - 강제로 이벤트를 호출할 때
    - 네임스페이스로 식별력을 부여하면 하나의 요소에 연결된 여러 개의 이벤트 처리 함수 중에서 특정한 함수만을 호출할 수 있다.
    - 이벤트 객체를 통해 namespace 속성을 제공받을 수 있다.

jQuery 이벤트 설정(8)

- 예제 11-09

[예제 11-09 : 네임스페이스 활용]

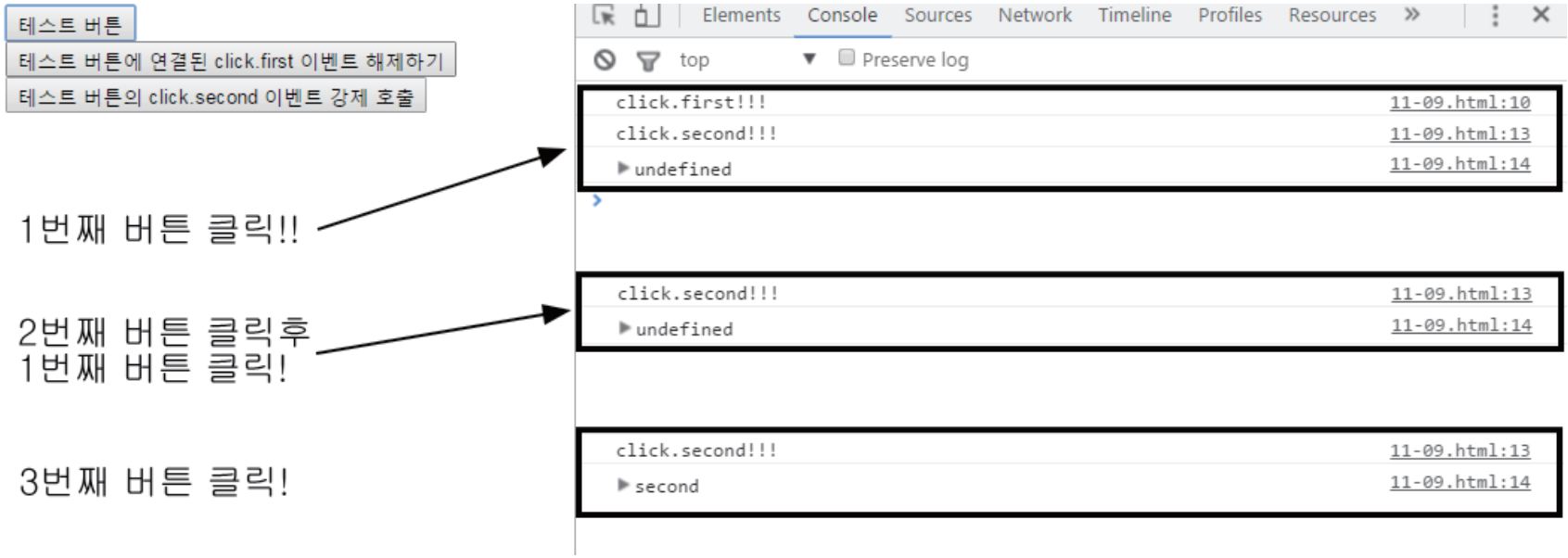
```
01: <!DOCTYPE html>
02: <html lang="">
03: <head>
04: <meta charset="utf-8">
05: <title>11-09</title>
06: <script src="http://code.jquery.com/jquery-3.1.0.js"></script>
07: <script type="text/javascript">
08: $(document).ready(function() {
09:   $("#test1").on("click.first", function(e) {
10:      console.log("click.first!!!");
11:   });
```

- "click.first", "click.second"와 같은 이벤트 타입 사용
- 도트(.) 기호 다음이 네임스페이스

```
12:   $("#test1").on("click.second", function(e) {
13:      console.log("click.second!!!");
14:      console.dir(e.namespace);
15:   });
16:   $("#unbinding").on("click", function(e) {
17:      $("#test1").off("click.first");
18:   });
19:   $("#trigger").on("click", function() {
20:      $("#test1").trigger("click.second");
21:   });
22: });
23: </script>
24: </head>
25: <body>
26:   <button id="test1">테스트 버튼</button> <br />
27:   <button id="unbinding">테스트 버튼에 연결된  click.first 이벤트 해제하기</button>
28:   <button id="trigger">테스트 버튼의 click.second 이벤트 강제 호출</button>
29: </body>
30: </html>
```

jQuery 이벤트 설정(9)

- 예제 11-09 실행 결과



이벤트 객체 정보(1)

이벤트 객체

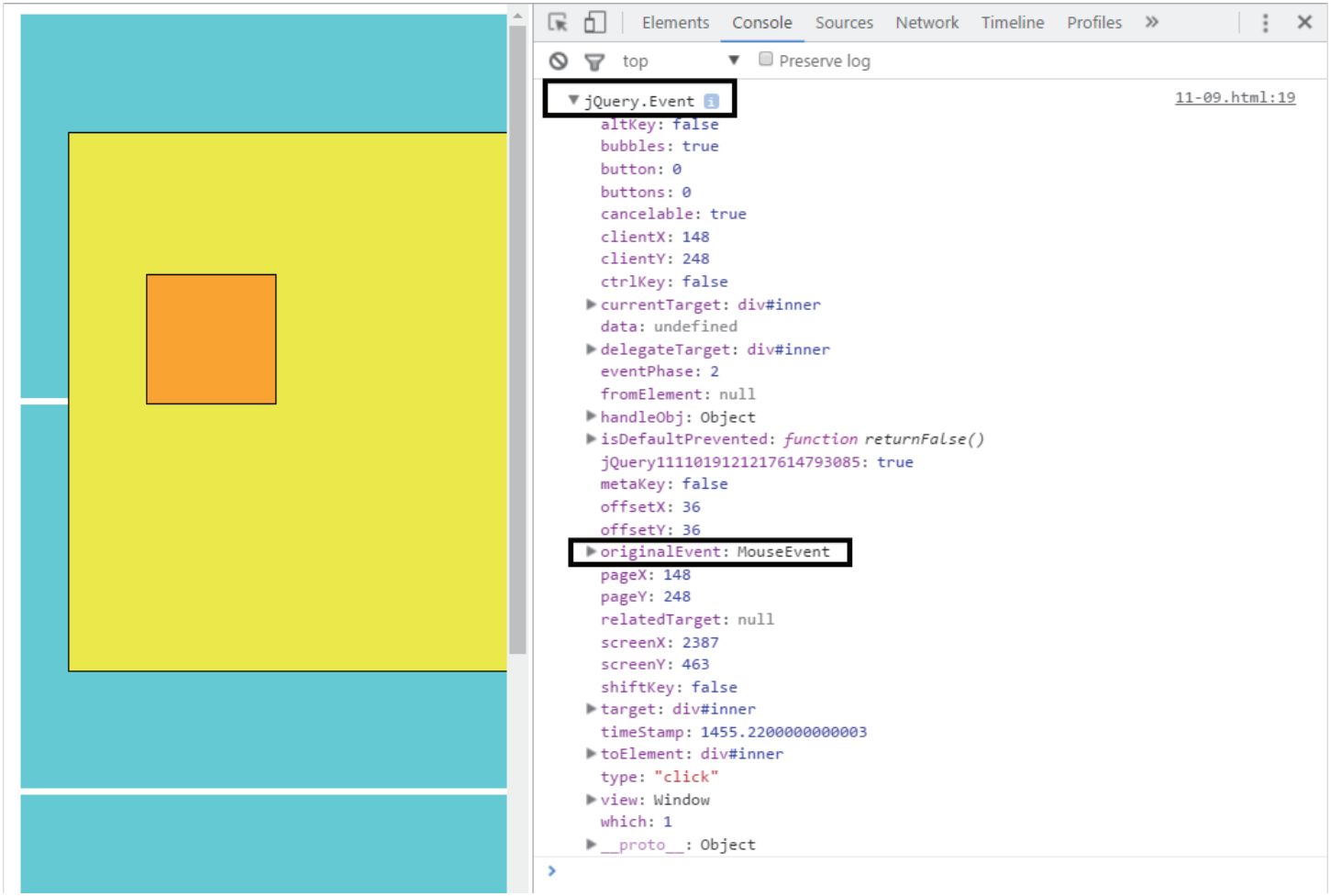
- jQuery 이벤트 처리 함수의 첫번째 인자로 전달되는 객체
- 모든 브라우저에서 이용할 수 있도록 W3C 표준에 따르도록 추상화
- 예제 11-10 : 예제 10-14를 복사해 볼드체 부분의 코드 추가

```
18:   $("#inner").on("click", function(e) {
19:      console.log(e);
20:   });
```

- 실행한 후 브라우저에서 div#inner 클릭
- jQuery.Event 객체 확인!

이벤트 객체 정보(2)

- 예제 11-10 실행 결과



이벤트 객체 정보(3)

- 이벤트 객체 공통 속성

[**표 11-02** : 이벤트 객체의 공통 속성]

속성명	설명
altKey	이벤트 발생 시에 ALT키를 눌렀는지를 true/false로 나타낸다.
ctrlKey	이벤트 발생 시에 CTRL키를 눌렀는지를 true/false로 나타낸다.
shiftKey	이벤트 발생 시에 SHIFT키를 눌렀는지를 true/false로 나타낸다.
metaKey	이벤트 발생 시에 메타키를 눌렀는지를 true/false로 나타낸다. 윈도우 운영체제에서는 Window키이고, OSX에서는 Command키이다.
type	이벤트 타입을 나타냄. click 이벤트라면 "click" 문자열을 리턴한다.
target	이벤트가 발생된 객체를 가리킨다.
currentTarget	이벤트 버블링 시의 현재 객체를 가리킨다. 자세한 내용은 이벤트 전파를 다루면서 학습한다.
delegateTarget	이벤트 위임 처리 시에 이 속성을 사용한다. 이벤트를 위임 처리하는 객체를 가리킨다. 자세한 내용은 이벤트 위임 처리를 다루면서 학습한다.
data	이벤트 연결 시에 지정한 데이터값. '2.4 이벤트 데이터 전달'에서 다룬 내용이다.
eventPhase	이벤트 진행 단계값. 자세한 내용은 이벤트 전파를 다루면서 학습한다.
bubbles	이벤트 버블링을 일으키는지를 true/false로 나타낸다.
cancelable	기본 이벤트 실행을 취소할 수 있는 상태인지를 나타내는 값. 이 값이 true면 preventDefault( ) 메서드를 호출해 기본 이벤트 실행을 중단할 수 있다.
timestamp	가장 직전에 발생한 이벤트로부터 현재 이벤트 발생 시점까지 흐른 시간값이다. ms 단위로 측정한다.
which	키보드, 마우스 이벤트가 발생했을 때 눌러진 버튼, 키보드의 값을 나타낸다. 키보드는 아스키 코드값이 넘어오며, 마우스인 경우는 왼쪽:1, 중간휠:2, 오른쪽:3의 값이 전달된다. button 속성과 값이 조금 다르다.

이벤트 객체 정보(4)

- 이벤트 객체의 마우스, 키보드 관련 속성

[**표 11-03** : 이벤트 객체의 마우스 관련 속성]

속성명	설명
button	클릭된 마우스 버튼값(0:왼쪽 버튼, 1:휠, 2:오른쪽 버튼)
buttons	남아 있는 클릭된 마우스 버튼 수. 여러 개 버튼을 동시에 클릭했다가 하나씩 눌렀던 마우스 버튼을 떼어보면 값의 변화를 알 수 있다.
clientX, clientY	뷰포트(ViewPort) 즉, 브라우저에서 문서가 나타나는 화면 영역에서의 이벤트가 발생된 지점의 좌표값.
offsetX, offsetY	이벤트가 발생된 HTML 요소상에서의 좌표값.
pageX, pageY	HTML 문서 영역에서의 좌표값
screenX, screenY	스크린 영역에서의 좌표값

[**표 11-04** : 키보드 이벤트 발생 시 속성]

속성명	설명
keyCode	키보드 이벤트(keyup, keydown) 발생 시 눌렀던 키의 아스키 코드값을 나타낸다.
charCode	키보드 이벤트(keypress) 발생 시 눌렀던 키의 아스키 코드값을 리턴한다. keyup, keydown 이벤트 발생 시에는 charCode값을 이용할 수 없다.

이벤트 객체 정보(5)

- 이벤트 객체의 메서드

[**표 11-05** : 이벤트 객체가 제공하는 메서드]

메서드명	설명
preventDefault( )	기본 이벤트 실행을 중지시킨다. 자세한 내용은 다음 절 '기본 이벤트와 이벤트 전파'에서 다룬다.
stopPropagation( )	상위 요소로의 이벤트 전파를 중단시킨다. 자세한 내용은 다음 절 '기본 이벤트와 이벤트 전파'에서 다룬다.
stopImmediatePropagation( )	상위 요소로의 이벤트 전파를 중단시키고 현재 DOM의 HTML 요소의 다른 이벤트로의 전파도 중단시킨다.
isDefaultPrevented( )	preventDefault( ) 메서드가 호출되었는지를 true/false로 나타낸다.
isPropagationStopped( )	stopPropagation( ) 메서드가 호출되었는지를 true/false로 나타낸다.
isImmediatePropagationStopped( )	stopImmediatePropagation( ) 메서드가 호출되었는지를 true/false로 나타낸다.

기본 이벤트와 이벤트 전파(1)

기본 이벤트

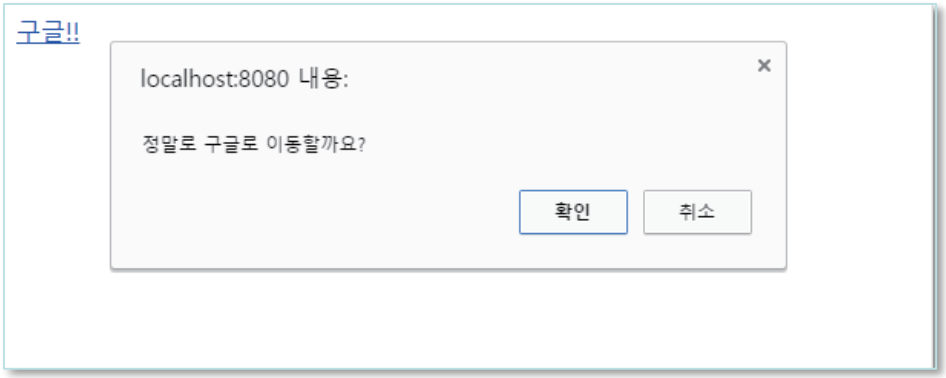
- 이미 HTML 문서나 요소에 어떤 기능을 실행하도록 정의되어 있는 기능
- 대표적인 기본 이벤트
  - <a> 요소를 클릭했을 때 href 특성의 경로로 페이지가 이동함
  - 브라우저 화면을 마우스 오른쪽 클릭했을 때 내장 컨텍스트 메뉴가 나타남
  - <form> 요소 내부의 submit 버튼을 클릭했을 때 <form> 요소의 action 특성에 지정된 경로로 method 특성에 지정된 방식으로 전송함.
  - <input type="text" ... /> 요소에 키보드를 누르면 입력한 문자가 텍스트박스에 나타남
- 기본 이벤트가 편리하긴 하지만 때로는 기본 이벤트 실행을 막아야할 때도 있음

기본 이벤트와 이벤트 전파(2)

- 예제 11-11
  - <a /> 클릭했을 때 confirm() 통해 취소를 누르면 페이지 이동 중단
  - document의 컨텍스트 메뉴 실행 중단

```
21: <body>
22:   <a id="google" href="http://google.com">구글!!</a>
23: </body>
```

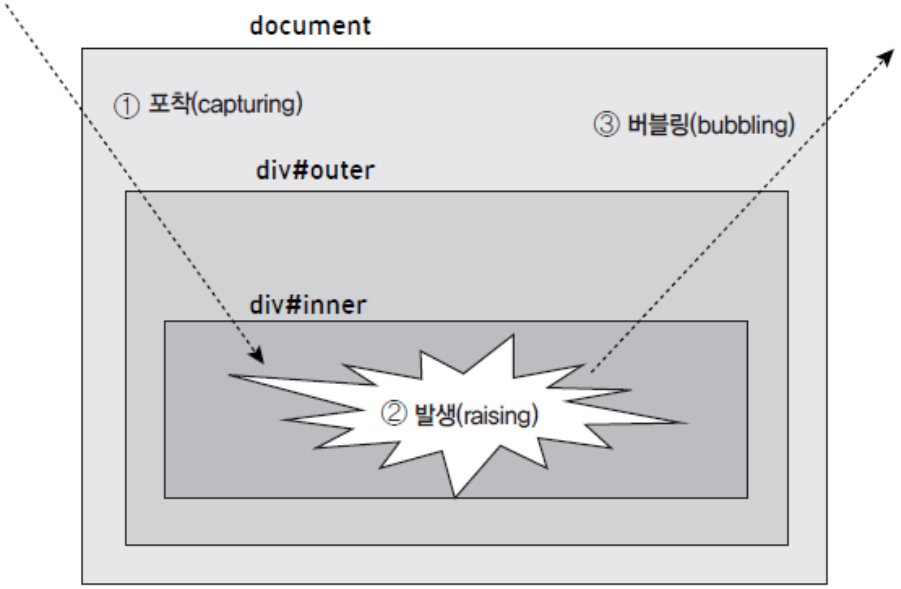
```
08: $(document).ready(function() {
09:   $("#google").on("click", function(e) {
10:      if (confirm("정말로 구글로 이동할까요?") == false) {
11:         e.preventDefault();
12:      }
13:   });
14:
15:   $(document).on("contextmenu", function(e) {
16:      e.preventDefault();
17:   });
18: });
```



기본 이벤트와 이벤트 전파(3)

이벤트 전파

- 이벤트 전파의 3단계
  - 포착(Capture) : HTML 문서 밖에서 이벤트가 발생한 요소를 찾아들어가는 단계
  - 발생(Raise) : 이벤트에 연결된 함수를 호출시켜 이벤트를 발생시키는 단계
  - 버블링(Bubble) : 이벤트가 발생한 요소로부터 상위 요소로 거슬러 올라가면서 동일한 이벤트 처리 함수를 호출시키는 단계
- jQuery 이벤트에서는 포착단계에서는 아무런 기능이 수행되지 않는다.

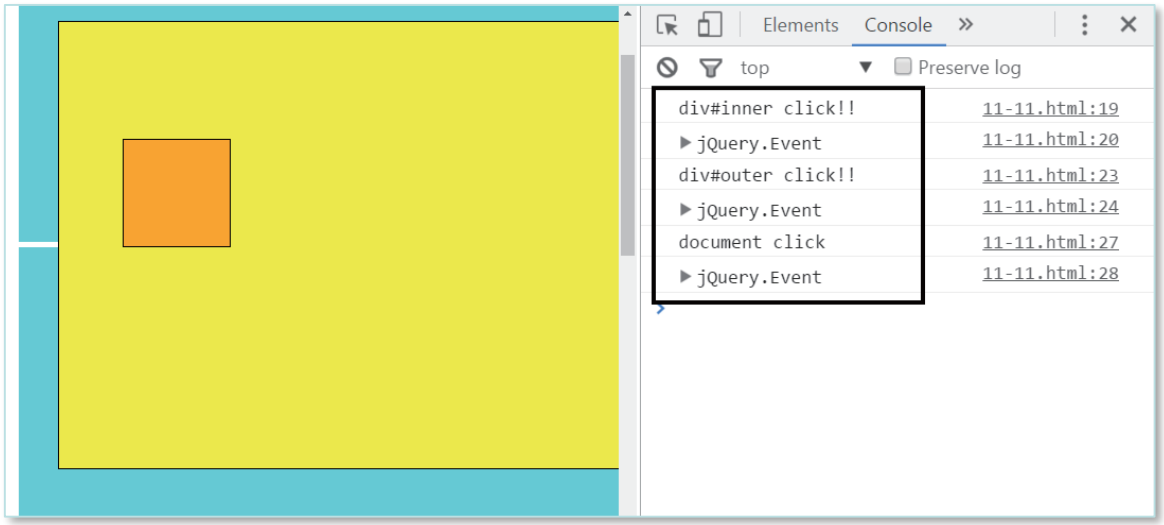


기본 이벤트와 이벤트 전파(4)

- 예제 11-12 : 예제 10-14를 이용해 작성

```
18:  $("#inner").on("click", function(e) {
19:      console.log("div#inner click!!");
20:      console.dir(e);
21:  });
22:  $("#outer").on("click", function(e) {
23:      console.log("div#outer click!!");
24:      console.dir(e);
25:  });
26:  $(document).on("click", function(e) {
27:      console.log("document click");
28:      console.dir(e);
29:  });
```

예제 11-12 실행 후 #inner 클릭



- #inner click 이벤트 발생 후 버블링하면서 #outer click, document click 이벤트가 모두 발생한다.
- 예외적인 경우가 아니라면 일반적으로 이벤트 버블링 막기를 원할 것이다.

기본 이벤트와 이벤트 전파(5)

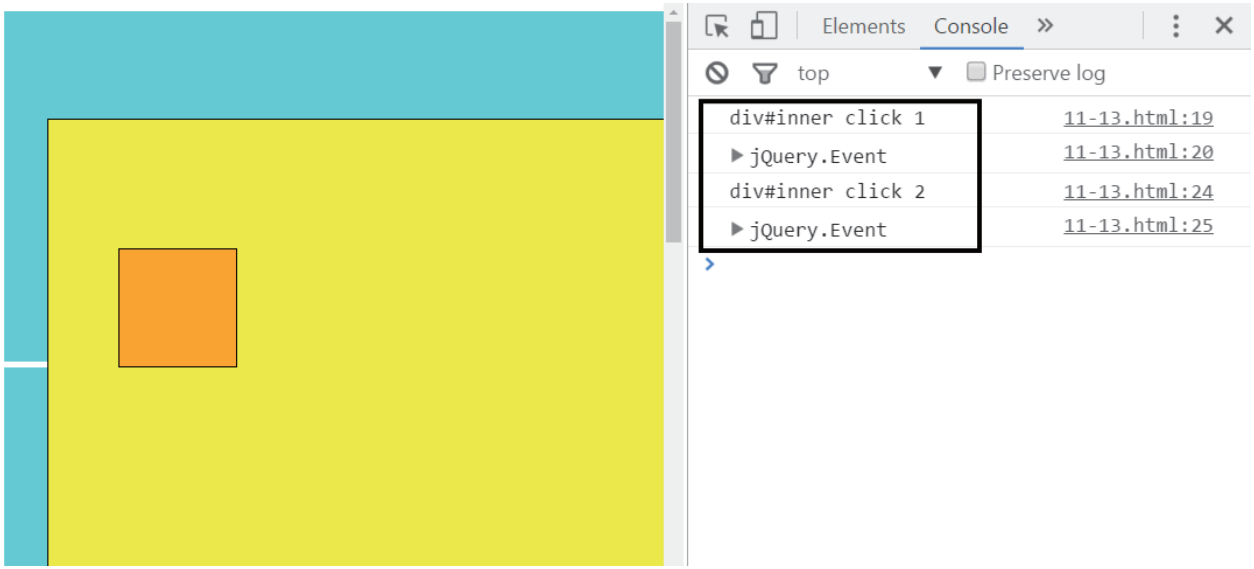
- stopPropagation()
  - 이벤트 전파를 중지시킴
- stopImmediatePropagation()
  - 이벤트 실행을 중지시킴. 하지만 stopPropagation() 메서드와는 약간의 차이
  - 동일 요소, 동일 이벤트에 여러 개의 함수가 연결되어 있을 때
    - stopPropagation() 메서드는 상위 요소로의 이벤트 전파만 막고 동일 요소의 나머지 이벤트에 대해서는 이벤트 전파를 막지 않는다
    - stopImmediatePropagation() 메서드는 만일 동일 요소의 나머지 이벤트에 대해서도 전파를 막는다.

예제 11-14

```
03:   $("#inner").on("click", function(e) {
04:      console.log("div#inner click 1");
05:      console.dir(e);
06:      e.stopPropagation();
07:   });
08:   $("#inner").on("click", function(e) {
09:      console.log("div#inner click 2");
10:      console.dir(e);
11:      e.stopPropagation();
12:   });
13:   $("#outer").on("click", function(e) {
14:      console.log("div#outer click");
15:      console.dir(e);
16:      e.stopPropagation();
17:   });
18:   $(document).on("click", function(e) {
19:      console.log("document click");
20:      console.dir(e);
21:      e.stopPropagation();
22:   });
```

기본 이벤트와 이벤트 전파(6)

- 예제 11-14 실행 결과
  - 실행 후 #inner 클릭!!



  - stopPropagation() 메서드를 stopImmediatePropagation() 메서드로 변경하면 "div#innder click2"는 호출되지 않음

기본 이벤트와 이벤트 전파(7)

- 이벤트 발생 단계와 버블링 단계의 이벤트 객체 차이
  - eventPhase : 1-Capture, 2-Raise, 3-Bubble4
  - target : Raise된 HTML 요소
  - currentTarget : 지금 호출되고 있는 이벤트의 HTML 요소

발생(raising) 단계 이벤트 객체

```
▼jQuery.Event
  altKey: false
  bubbles: true
  button: 0
  buttons: 0
  cancelable: true
  clientX: 173
  clientY: 243
  ctrlKey: false
▶ currentTarget: div#inner
  data: undefined
▶ delegateTarget: div#inner
  eventPhase: 2
  fromElement: null
▶ handleObj: Object
▶ isDefaultPrevented: function returnFalse()
  jQuery111100618655413342612 54: true
  metaKey: false
  offsetX: 61
  offsetY: 31
▶ originalEvent: MouseEvent
  pageX: 173
  pageY: 243
  relatedTarget: null
  screenX: 258
  screenY: 370
  shiftKey: false
▶ target: div#inner
  timeStamp: 1343.0900000000001
▶ toElement: div#inner
  type: "click"
▶ view: Window
```

버블링(bubbling) 단계 이벤트 객체

```
▼jQuery.Event
  altKey: false
  bubbles: true
  button: 0
  buttons: 0
  cancelable: true
  clientX: 173
  clientY: 243
  ctrlKey: false
▶ currentTarget: div#outer
  data: undefined
▶ delegateTarget: div#outer
  eventPhase: 3
  fromElement: null
▶ handleObj: Object
▶ isDefaultPrevented: function returnFalse()
  jQuery111100618655413342612 54: true
  metaKey: false
  offsetX: 61
  offsetY: 31
▶ originalEvent: MouseEvent
  pageX: 173
  pageY: 243
  relatedTarget: null
  screenX: 258
  screenY: 370
  shiftKey: false
▶ target: div#inner
  timeStamp: 1343.0900000000001
▶ toElement: div#inner
  type: "click"
▶ view: Window
```

이벤트 위임 처리(1)

이벤트 버블링은 막아야만 하나?

- 때로는 활용하는 경우도 있다.
- 전자 정부 표준 프레임워크 조회 화면 중 한 사례

시스템이력 조회 --선택하세요-- 조회 등록

번호	이력 ID	시스템명	이력구분	등록자	등록일자	상세보기
1	HT_20090430070158937	테스트 시스템	소프트웨어삭제	업무담당자	2009-04-30	
2	HT_20090429010011834	테스트 시스템	소프트웨어삭제	마스터	2009-04-29	
3	HT_20090428112751045	테스트 시스템	소프트웨어설치	박상수	2009-04-28	
4	HT_20090415022511968	공통시스템	소프트웨어설치	업무담당자	2009-04-15	
5	HT_20090415025752093	공통시스템	소프트웨어패치	업무담당자	2009-04-15	
6	HT_20090415030018687	공통시스템	소프트웨어설치	업무담당자	2009-04-15	
7	HT_20090415030038500	공통시스템	소프트웨어삭제	업무담당자	2009-04-15	

- 조회하는 데이터 개수가 매번 다를텐데 조회할 때마다 매번 상세보기 버튼에 이벤트를 연결하는 것은 비효율적이다.

이벤트 위임 처리(2)

- 이벤트 위임 처리
  - 이벤트 버블링을 활용해 이벤트가 발생하는 요소가 상위 요소에 이벤트 처리를 위임한다.
  - 이벤트가 발생하는 요소가 없어도 이벤트를 미리 설정할 수 있다. 상위요소에 이벤트를 위임 처리하기 때문이다.
  - 두가지 메서드. on(), delegate()

- $(상위요소).delegate(selector, event_type , handler_function)
- $(상위요소).on(event_type, selector, handler_function)

상위요소	이벤트를 위임 처리하는 상위 HTML 요소
selector	이벤트를 발생시키는 자식 요소를 나타낼 수 있는 선택자
event_type	click, keyup과 같은 이벤트명
handler_function	이벤트 위임으로 실행할 이벤트 처리 함수

이벤트 위임 처리(3)

- 이벤트 위임 예제
- 예제 11-15 : HTML 마크업

```
06: <style>
07:     #nations > thead > tr { background-color:purple; color:yellow; }
08:     #nations > thead > tr > th { padding :0px 1px 0px 0px; width:150px; }
09:     #nations > tbody > tr > td { border-bottom: solid 1px gray; }
10: </style>
```

```
19:   지역 선택 :
20:   <select id="region">
21:      <option value="all" selected>전체 정보 보기</option>
22:      <option value="asia">아시아</option>
23:      <option value="america">미주</option>
24:      <option value="europe">유럽</option>
25:      <option value="oceania">대양주</option>
26:      <option value="africa">아프리카</option>
27:   </select>
28:   <button id="inquery">조회</button>
29:     <table id="nations">
30:         <thead>
31:             <tr><th>국가명</th><th>수도</th><th>상세보기</th></tr>
32:         </thead>
33:         <tbody id="list">
34:
35:         </tbody>
36:     </table>
37: </body>
38: </html>
```

이벤트 위임 처리(4)

- 예제 11-16 : 샘플 데이터, Template 작성

```
01: <script type="text/javascript">
02: $(document).ready(function() {
03:     var data = [
04:             { no:1,  name : "미국", capital : "워싱턴DC", region:"america" },
05:             { no:2,  name : "프랑스", capital : "파리", region:"europe" },
06:             { no:3,  name : "영국", capital : "런던", region:"europe" },
07:             { no:4,  name : "중국", capital : "베이징", region:"asia" },
08:             { no:5,  name : "태국", capital : "방콕", region:"asia" },
09:             { no:6,  name : "모로코", capital : "라바트", region:"africa" },
10:             { no:7,  name : "라오스", capital : "비엔티안", region:"asia" },
11:             { no:8,  name : "베트남", capital : "하노이", region:"asia" },
12:             { no:9,  name : "피지", capital : "수바", region:"oceania" },
13:            { no:10,  name : "솔로몬 제도", capital : "호니아라", region:"oceania" },
14:             { no:11,  name : "자메이카", capital : "킹스턴", region:"america" },
15:             { no:12,  name : "나미비아", capital : "빈트후크", region:"africa" },
16:             { no:13,  name : "동티모르", capital : "딜리", region:"asia" },
17:             { no:14,  name : "멕시코", capital : "멕시코시티", region:"america" },
18:             { no:15,  name : "베네수엘라", capital : "카라카스", region:"america" },
19:             { no:16,  name : "서사모아", capital : "아피아", region:"oceania" }
20:         ];
21:
22:     var template = "<tr><td>${name}</td><td>${capital}</td>" +
23:         "<td><button class='detail' data-no='${no}'>상세보기</button></td></tr>";
24:
25: });
```

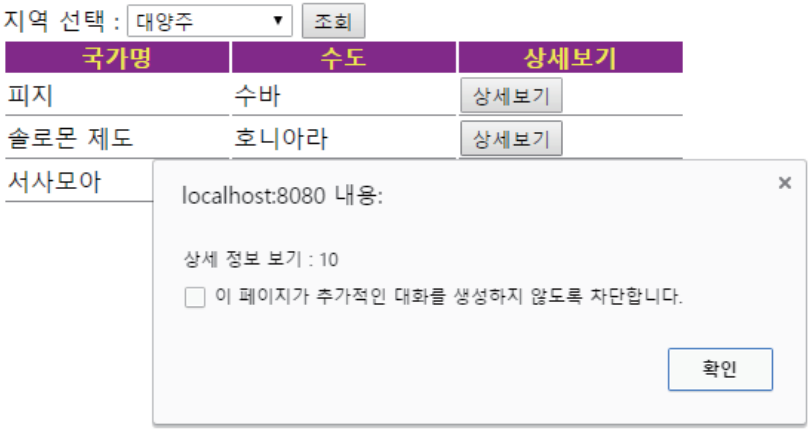
이벤트 위임 처리(5)

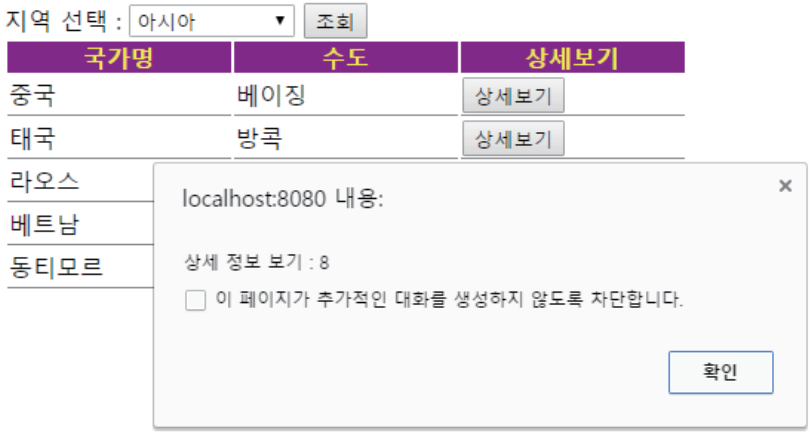
- 예제 11-17

```
07:         //이곳에 나머지 코드를 배치함.
08:         function filterData(region) {
09:             var result = data.filter(function(v, index) {
10:                 if (v.region == region) {
11:                     return true;
12:                 }
13:             });
14:             return result;
15:         }
16:
17:         $("#list").delegate("button.detail", "click", function(e) {
18:             var cno = $(this).attr("data-no");
19:
20:             alert("상세 정보 보기 : " + cno);
21:         });
22:
23:         $("#inquery").click(function() {
24:             var arr, str, s;
25:             var sel = $("#region  option:selected").val();
26:
27:             if (sel == "all") {
28:                 arr = data;
29:             } else {
30:                 arr = filterData(sel);
31:             }
32:
33:             arr.forEach(function(item, index) {
34:                 s = template;
35:                 s = s.replace("${no}", item.no);
36:                 s = s.replace("${name}", item.name);
37:                 s = s.replace("${capital}", item.capital);
38:                 str += s;
39:             });
40:             $("#list").empty();
41:             $("#list").append(str);
42:         });
43: });
44: </script>
```

이벤트 위임 처리(6)

- 예제 11-15~17 까지의 정리
  - 지역 정보 데이터를 필터링해 보여줌
  - 예제 11-17의 17행 : 이벤트 위임 처리 코드. delegate 메서드 사용
  - 예제 11-17의 23~42행 : 화면 UI를 생성하는 코드. 조회버튼을 클릭할 때마다 기존 내용을 삭제하고 다시 출력하기 때문에 '상세보기' 버튼은 매번 만들어진다. 하지만 이벤트는 매번 연결할 필요 없음

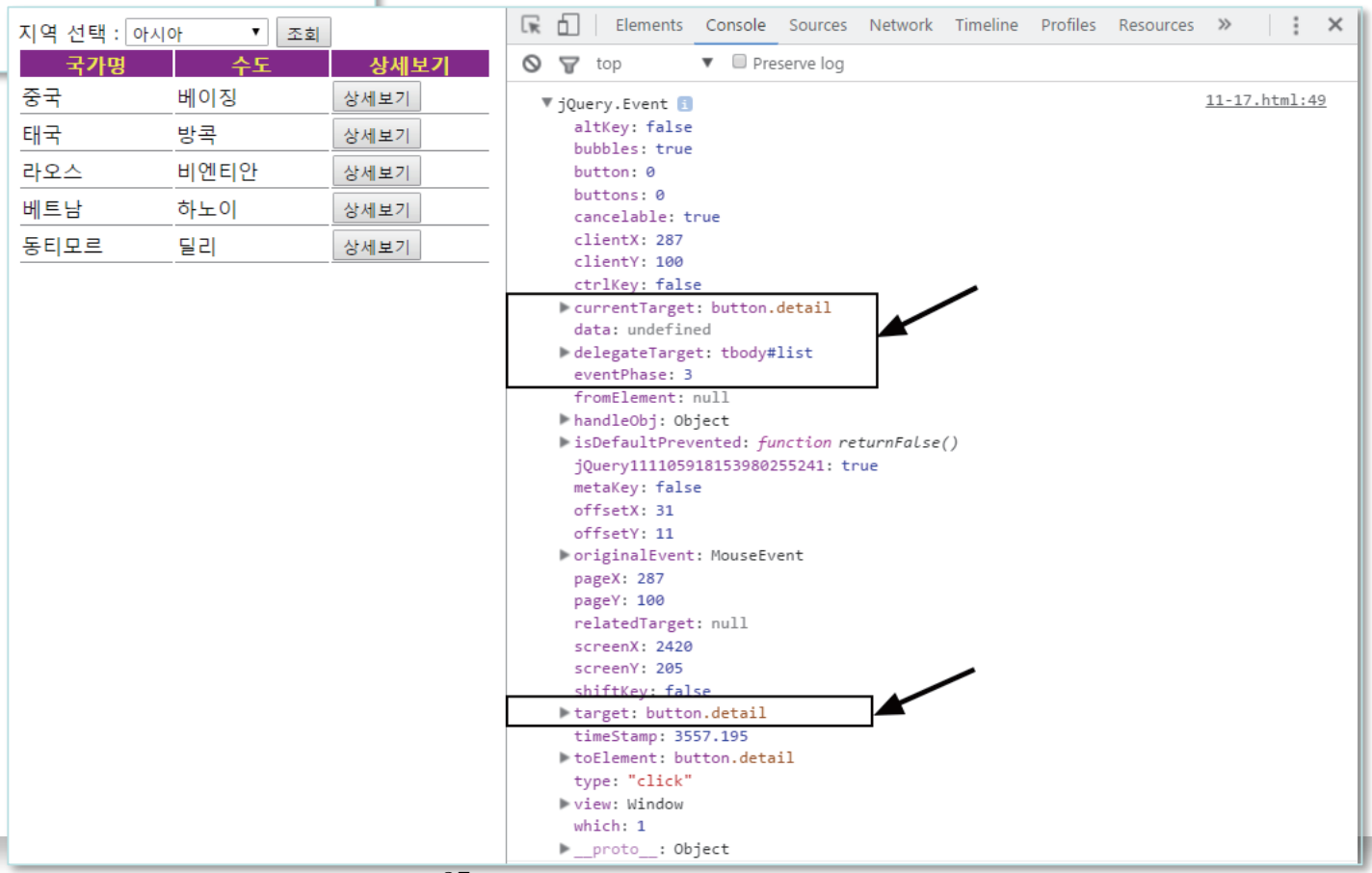




이벤트 위임 처리(7)

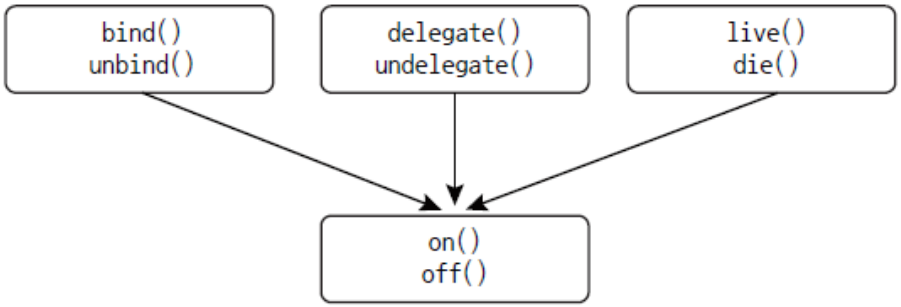
- 이벤트 위임 처리할 때의 이벤트 객체

```
$("#list").delegate("button.detail", "click", function(e) {
    var cno = $(this).attr("data-no");
    console.dir(e);
    alert("상세 정보 보기 : " + cno);
});
```



이벤트 위임 처리(8)

- 이벤트 메서드들간의 관계
  - bind(), delegate(), live() 메서드가 on()으로 통합되었다.
  - bind(), delegate() 모두 내부적으로 on() 메서드를 호출하므로 이벤트 위임 처리를 위해 delegate()를 사용하는 것도 나쁘지 않다.



[**표 10-06** : 이벤트 연결 메서드 비교]

개별 메서드
$("#a").bind("click", function(e) { }); → $("#a").on("click", function(e) { });
$("#list").delegate("button.detail", "click", function(e) { }); → $("#list").on("click", "button.detail", function(e) { });
$("button.detail").live("click", function(e) { }); → $(document).delegate("button.detail", "click", function(e) { });

* live() 메서드는 1.9버전부터 지원 중단

이벤트 위임 처리(9)

- 이벤트 위임 처리의 성능
  - 많은 수의 요소에 이벤트를 설정해야 하는 경우 이벤트 위임 처리는 일반적인 이벤트 연결보다 성능이 좋다. j

Test runner

Done. Ready to run again.

Run again

Testing in Chrome 49.0.2623.87 32-bit on Windows Server 2008 R2 / 7 64-bit

Test		Ops/sec

direct	`$('.target').on('click', someFunc);`
	
	`var someFunc = function(event) {`
	`  $this = $(this);`
	`  $this.hide();`
	`  };`
		183,472
		±1.69%
		30% slower

delegate document	`$(document).on('click', '.target', someFunc);`
	
	`var someFunc = function(event) {`
	`  $this = $(this);`
	`  $this.hide();`
	`  };`
		79,484
		±2.54%
		70% slower

delegate parent	`$('#parent').on('click', '.target', someFunc);`
	
	`var someFunc = function(event) {`
	`  $this = $(this);`
	`  $this.hide();`
	`  };`
		261,452
		±15.92%
		fastest

마우스 관련 이벤트(1)

마우스 이벤트 종류

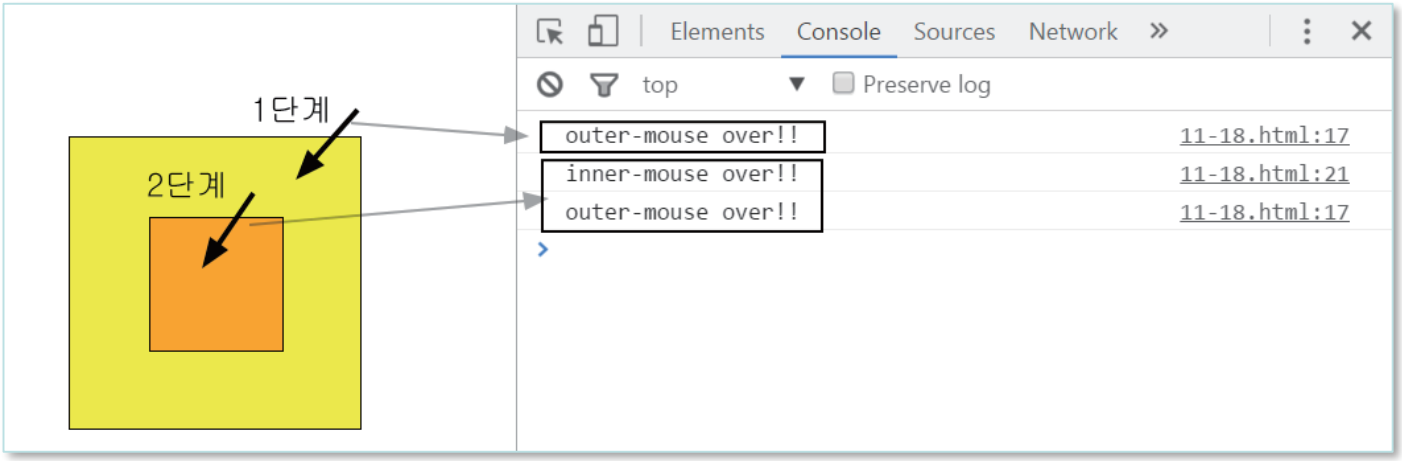
[**표 11-07** : 마우스 관련 이벤트]

이벤트	설명
click	마우스를 클릭할 때 발생하는 이벤트
dblclick	마우스를 더블클릭할 때 발생하는 이벤트
mousedown	마우스 버튼을 누를 때 발생하는 이벤트
mouseup	마우스 버튼을 눌렀다 뗄 때 발생하는 이벤트
mousemove	마우스를 움직일 때 발생하는 이벤트
mouseenter	마우스가 HTML 요소의 경계 안쪽으로 넘어갈 때 발생하는 이벤트
mouseleave	마우스가 HTML 요소의 경계 밖으로 넘어갈 때 발생하는 이벤트
mouseover	마우스가 HTML 요소를 벗어날 때 발생하는 이벤트
mouseout	마우스가 HTML 요소 안으로 들어올 때 발생하는 이벤트

마우스 관련 이벤트(2)

- mouseover, mouseout 대신에 mouseenter, mouseleave 이벤트 사용을 권장한다.
  - mouseover, mouseout은 버블링하는 이벤트이기 때문에…
  - 예제 11-18 실행 결과

```
16:  $("#outer").on("mouseover", function(e) {
17:     console.log("outer-mouse over!!");
18:  });
19:
20:  $("#inner").on("mouseover", function(e) {
21:     console.log("inner-mouse over!!");
22:  });
```

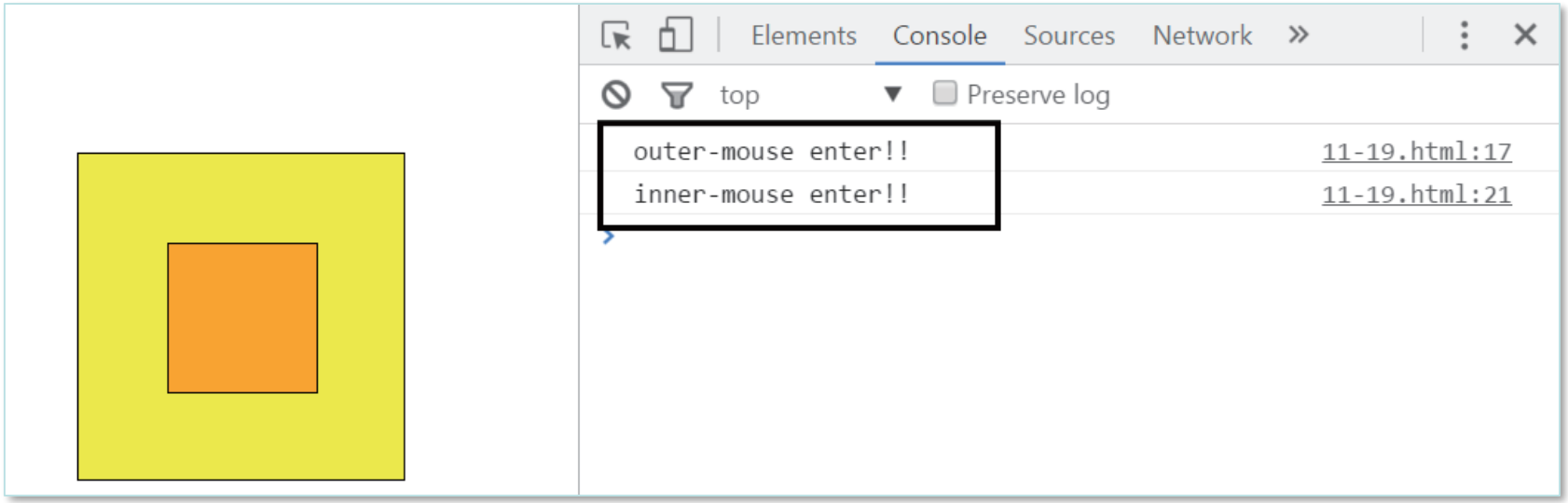


마우스 관련 이벤트(3)

- 예제 11-18의 코드를 mouseenter, mouseleave 이벤트로 변경

```
$("#outer").on("mouseenter", function(e) {
   console.log("outer-mouse enter!!");
});

$("#inner").on("mouseenter", function(e) {
   console.log("inner-mouse enter!!");
});
```



마우스 관련 이벤트(4)

- 예제 11-20

```
24:     <table id="nations">
25:         <thead>
26:             <tr><th>번호</th><th>국가명</th><th>수도</th></tr>
27:         </thead>
28:         <tbody>
29:             <tr><td>1</td><td>미국</td><td>워싱턴DC</td></tr>
30:             <tr><td>2</td><td>프랑스</td><td>파리</td></tr>
31:             <tr><td>3</td><td>영국</td><td>런던</td></tr>
32:             <tr><td>4</td><td>중국</td><td>베이징</td></tr>
33:             <tr><td>5</td><td>태국</td><td>방콕</td></tr>
34:             <tr><td>6</td><td>모로코</td><td>라바트</td></tr>
35:             <tr><td>7</td><td>라오스</td><td>비엔티안</td></tr>
36:         </tbody>
37:     </table>
```

```
15:   $("#nations > tbody > tr").mouseenter(function() {
16:       $(this).addClass("sel");
17:   }).mouseleave(function() {
18:       $(this).removeClass("sel");
19:   });
```

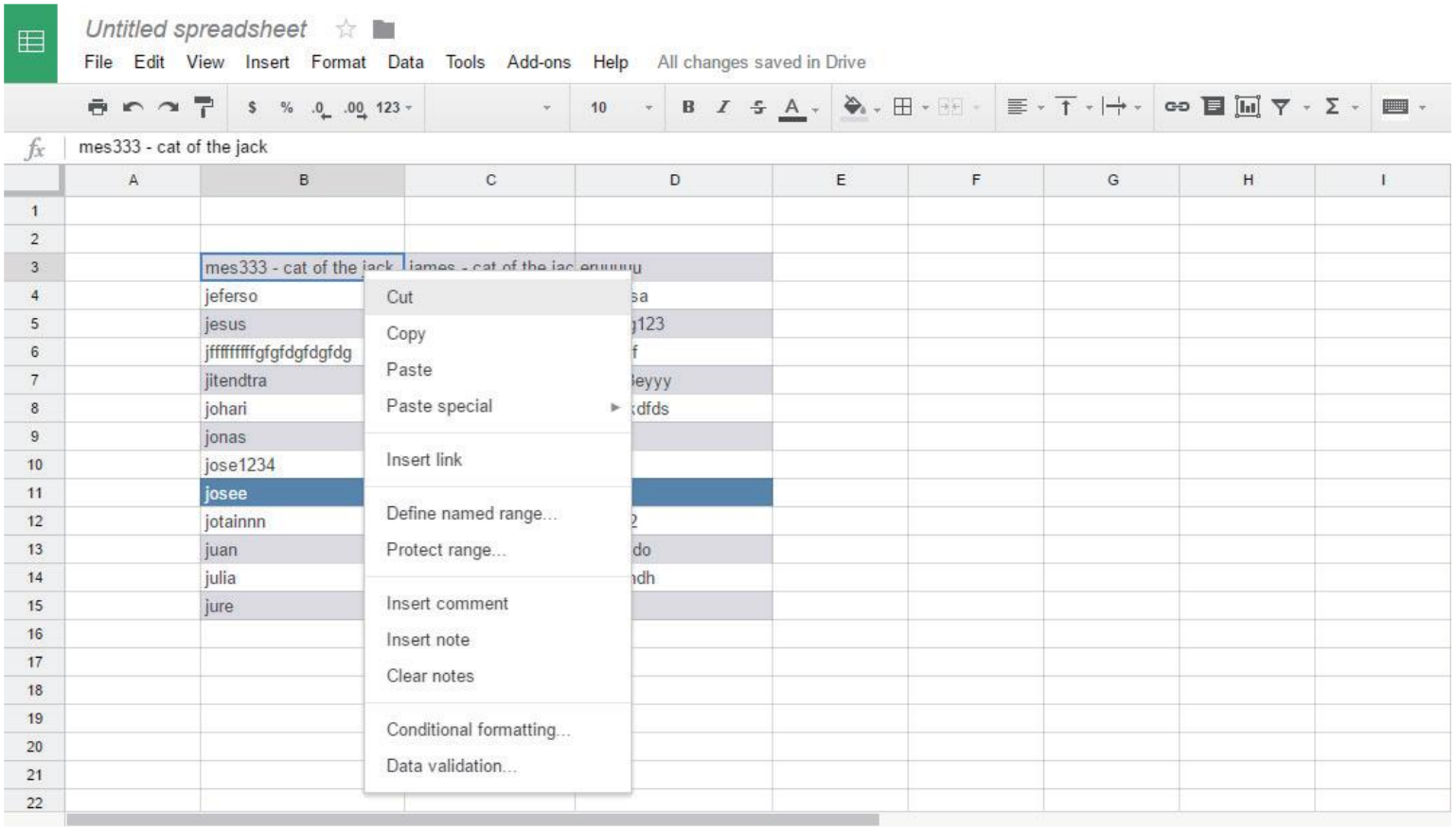
```
06: <style>
07:     #nations > thead > tr { background-color:purple; color:yellow; }
08:     #nations > thead > tr > th { padding :0px 1px 0px 0px; width:150px; }
09:     #nations > tbody > tr > td { border-bottom: solid 1px gray; }
10:     .sel { background-color : aqua; }
11: </style>
```

번호	국가명	수도
1	미국	워싱턴DC
2	프랑스	파리
3	영국	런던
4	중국	베이징
5	태국	방콕
6	모로코	라바트
7	라오스	비엔티안

마우스 포인터가 위치한 tr 열만 sel 클래스가 추가된다

마우스 관련 이벤트(5)

- 마우스 이벤트를 사용한 사용자 정의 컨텍스트 메뉴 작성
  - 대표 예 : 구글 스프레드시트 화면



마우스 관련 이벤트(6)

- 예제 11-21 : HTML 마크업

```
05: <style>
06:   div#popup { width:150px; font-size:9pt; z-index:999;
07:        border:solid 1px #DDDDDD; position:absolute; display:none;
08:        box-shadow: 0 0 5px #00EE3C;  }
09:   div#popup > div { background-color: white; text-align: center;
10:      padding:3px 5px 3px 5px; }
11:   div#popup > div:hover { background-color: aqua; cursor:pointer; }
12:   .dummy { height:400px; border:solid 1px #DDDDDD; }
13: </style>
```

```
22:   <div id="divA">김수한무<br />거북이와<br /> 두루미</div>
23:   <button id="btnA">테스트 버튼</button>
24:   <input type="text" id="name" value="" />
25:   <div class="dummy"></div>
26:   <div id="popup">
27:      <div data-cmd="cut">Cut</div>
28:      <div data-cmd="copy">Copy</div>
29:      <div data-cmd="delete">Delete</div>
30:      <div data-cmd="paste">Paste</div>
31:   </div>
```

마우스 관련 이벤트(7)

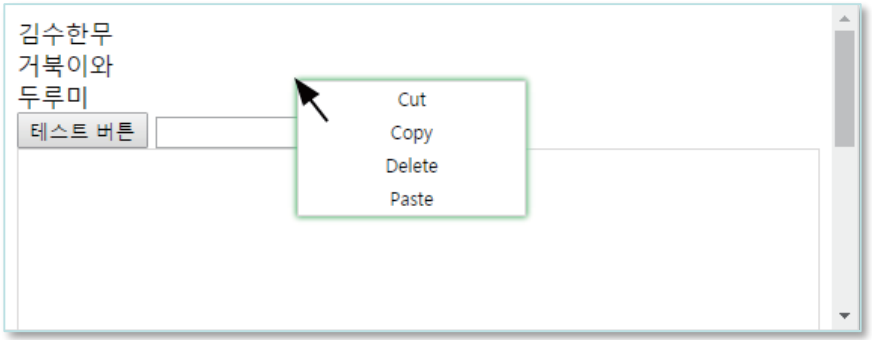
- 예제 11-22 : 기능 코드 작성

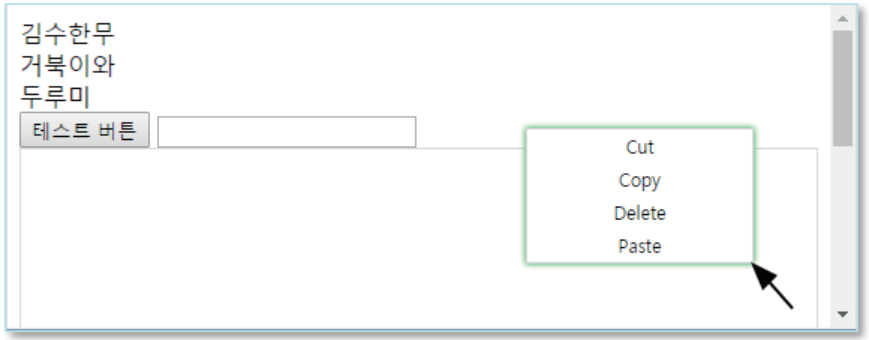
[예제 11-22]

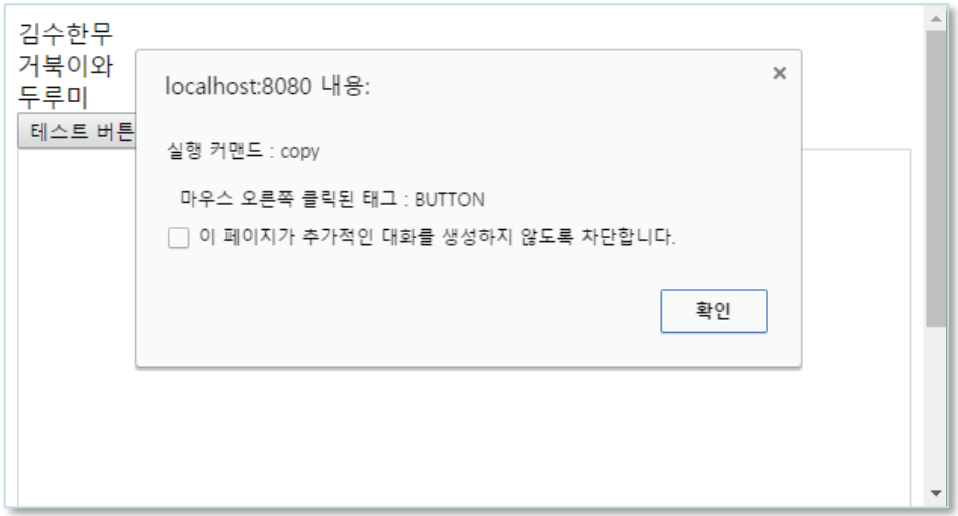
```
01: <script type="text/javascript">
02: $(document).ready(function () {
03:   var rightClicked;
04:   $(document).on("contextmenu", function(e) {
05:      //마우스 오른쪽 클릭된 요소 파악
06:      rightClicked = e.target;
07:      //컨텍스트 메뉴 요소 선택
08:      var menu = $("#popup");
09:      //오른쪽 클릭에 대한 default event 실행 취소
10:      e.preventDefault();
11:      //마우스 오른쪽 클릭한 좌표에 컨텍스트 메뉴 배치
12:      var pageX = e.pageX;
13:      var pageY = e.pageY;
14:      menu.css({top: pageY , left: pageX});
15:      //컨텍스트 메뉴 크기,브라우저 화면 영역(viewport) 파악
16:      var mwidth = menu.width();
17:      var mheight = menu.height();
18:      var screenWidth = $(window).width();
19:      var screenHeight = $(window).height();
20:      //화면이 스크롤 된 경우를 위해 스크롤한 높이를 파악
21:      var scrTop = $(window).scrollTop();
22:      //화면에서 오른쪽 경계선, 아래쪽 경계선 근처에서 클릭했을 때
23:      //컨텍스트 메뉴를 안정적으로 보이도록 좌표를 다시 설정한 후 출력
24:      if(pageX+mwidth > screenWidth){
25:         menu.css({left:pageX-mwidth});
26:      }
27:      if(pageY+mheight > screenHeight+scrTop){
28:         menu.css({top:pageY-mheight});
29:      }
30:      menu.show();
31:   });
32:   //문서에서 클릭이벤트가 발생할 경우 컨텍스트 메뉴 닫음
33:   $(document).on("click", function(){
34:      $("#popup").hide();
35:   });
36:   //컨텍스트 메뉴를 클릭하면 커맨드 실행
37:   $("#popup > div").click(function(e) {
38:      var cmd = $(this).data("cmd");
39:      $(this).parent().hide();
40:      alert("실행 커맨드 : " + cmd + "\n\n" +
41:         "  마우스 오른쪽 클릭된 태그 : " + rightClicked.tagName);
42:   });
43: });
44: </script>
```

마우스 관련 이벤트(8)

- 예제 11-22 실행 결과







키보드 관련 이벤트(1)

- 키보드관련 이벤트는 웹 애플리케이션에서 자주 사용되는 이벤트
  - 이미 예제 07-15에서 사용해본 keyup 이벤트

국가명 : 독

번호	국가명	수도
1	미국	워싱턴DC
3	영국	런던
4	중국	베이징
5	태국	방콕

[**표 11-08**: 키보드 관련 이벤트]

이벤트	설명
keydown	키보드를 누를 때 발생하는 이벤트
keypress	keydown 이후 키와 문자가 연결되었을 때 발생하는 이벤트
keyup	키보드를 눌렀다가 뗄 때 발생하는 이벤트
compositionstart	IME Mode에서 keydown한 이후에 발생하는 이벤트로 글자 조합이 시작되는 이벤트
compositionupdate	IME Mode에서 keydown한 이후에 발생하는 이벤트로 글자 조합이 진행 중인 이벤트
compositionend	IME Mode에서 keydown한 이후에 발생하는 이벤트로 글자 조합이 완료되는 이벤트

키보드 관련 이벤트(2)

키보드 이벤트 발생 순서

키를 누른다.

keydown 이벤트 발생

keypress 이벤트 발생

문자열이 화면에 찍힘(기본 이벤트 : Default Event)

눌렀던 키를 뗀다

keyup 이벤트 발생

keyup, keydown 이벤트

- 화면상의 어느 키를 눌러도 발생하는 이벤트
- 화면에 표시되지 않는 키도 이벤트를 발생시킴

keypress 이벤트

- 화면에 글자가 완성되어 나타나지 않는 경우에는 이벤트를 발생시키지 않는다

키보드 관련 이벤트(3)

- 키보드의 한글 처리 문제
  - keypress 이벤트
    - 한글 입력시 문자가 완성되지 않은 상태이기 때문에 이벤트가 발생하지 않음.
  - keyup, keydown 이벤트
    - 문자가 아니라 키보드 자판마다 고유키가 주어진 것이다
    - 한글 '홍'을 입력할 때 'ghd'에 해당하는 keyCode 값이 이벤트 객체를 통해 전달됨
- 키보드 이벤트 흐름에서 문자열이 화면에 찍히는 것이 기본 이벤트(default event)이다
  - 기본 이벤트가 실행되기 전이라면 이벤트 객체의 preventDefault() 메서드를 호출하여 기본 이벤트 실행을 막아서 문자 입력을 막을 수 있다.
  - 한글에는 적용할 수 없다는 단점이 있음

키보드 관련 이벤트(4)

예제 11-23

```
34:   <input type="text" class="number" /> <br />
35:   <input type="text" class="number" /> <br />
36:   <input type="text" class="number" /> <br />
```

```
09:   $("input.number").keydown(function(e) {
10:         var chrs = [48,49,50,51,52,53,54,55,56,57];    //키보드 위쪽 0~9
11:         chrs = chrs.concat([96,97,98,99,100,101,102,103,104,105]);  //숫자 키패드
12:         chrs = chrs.concat([37,38,39,40]); //방향키(커서키)
13:         chrs = chrs.concat([13,8,9]);      //엔터, 백스페이스, 탭 키
14:         chrs = chrs.concat([45,46,36,35]); //ins,del,home,end
15:         chrs = chrs.concat([16,17,18,190]);   //shift, ctrl, alt, 소수점(.)
16:
17:         var k = e.keyCode;
18:         var isValid = false;
19:         for(var i=0; i < chrs.length; i++) {
20:             if (k == chrs[i]) {
21:                 isValid = true;
22:                 break;
23:             }
24:         }
25:
26:         if (isValid == false) {
27:             e.preventDefault();
28:         }
29:   });
```

키보드 관련 이벤트(5)

composition~ 이벤트

- 한글 IME 모드 입력시에 발생하는 이벤트
- 하지만 브라우저마다 이벤트 작동 방식이 조금씩 다르다.
  - IE 9, 10, 11은 한글 입력이 완전히 끝날 때까지(IME 모드가 해제될 때까지) compositionend가 발생되지 않음
  - 파이어폭스와 크롬은 한글 한 글자가 완성될 때마다 compositionend 이벤트가 발생
- 따라서 한글 처리는 입력된 이후에 처리하는 편이 더 쉽다.
  - 정규식을 이용해 치환하는 방법을 사용함.

키보드 관련 이벤트(6)

예제 11-24

- 한글과 몇몇 문자만 입력가능한 텍스트 박스

```
18:   <input type="text" class="korean" /> <br />
19:   <input type="text" class="korean" /> <br />
20:   <input type="text" class="korean" /> <br />
```

```
08: $(document).ready(function() {
09:       $("input.korean").on("keyup  keypress", function(e) {
10:       var pat = /[^ㄱ-힣\s\,\.\"\']/g;
11:       var v = $(this).val();
12:       $(this).val(v.replace(pat, ""));
13:   });
```

브라우저, 문서, 입력폼 이벤트(1)

브라우저, 문서, 입력폼 이벤트 종류

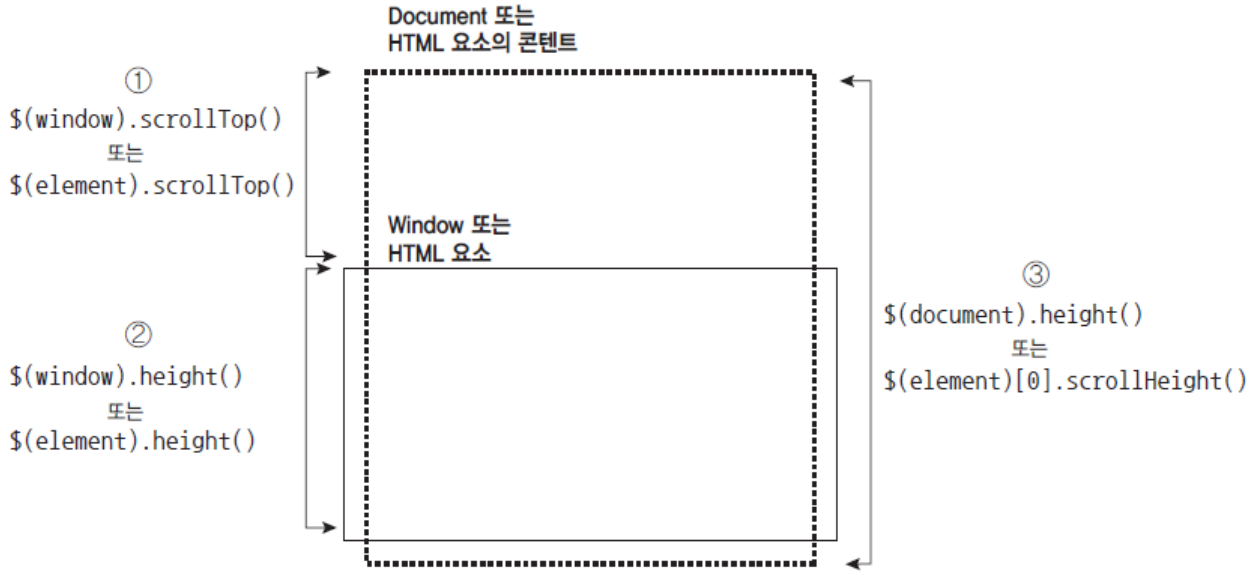
[**표 11-09**: 브라우저, 문서, 입력폼 관련 이벤트]

이벤트	설명
resize	브라우저의 크기가 변경될 때의 이벤트
scroll	브라우저, 문서, 요소에서 스크롤링했을 때의 이벤트
ready	DOM 요소들이 완전히 로드되었을 때의 이벤트
error	문서 요소의 완전한 로드에 실패했을 때의 이벤트. ex) 〈img /〉 요소의 src 특성에서 부여된 이미지를 로딩하지 못한 경우
load	문서의 로드가 완료될 때의 이벤트
unload	현재 보고 있는 페이지에서 다른 페이지로 이동할 때의 이벤트(deprecated)
focus	요소가 포커스를 획득했을 때의 이벤트
blur	요소가 포커스를 잃었을 때의 이벤트
change	입력폼 요소의 value 속성이 변경될 때의 이벤트. select박스, 체크박스, 라디오버튼에서는 마우스로 선택만 변경해도 즉시 이벤트가 발생되지만, 텍스트박스와 같은 경우는 포커스를 잃었을 때 발생한다.
select	사용자가 요소 내부의 텍스트를 선택했을 때의 이벤트. 〈input type="text" ... /〉, 〈textarea /〉 요소에서만 이벤트가 발생한다.
submit	〈form /〉 요소 내부의 submit 버튼이나 이미지 버튼을 클릭할 때 발생하는 이벤트이다.

브라우저, 문서, 입력폼 이벤트(2)

무한 스크롤 UI

- 최근 SNS 서비스들이 이 기법으로 화면을 구성
- 전통적인 페이징 기법은 페이지 이동 중에 데이터가 추가될 경우 데이터가 누락되거나 이미 보여진 데이터가 다시 나타나는 경우가 발생할 가능성이 있음



① + ② == ③

브라우저 스크롤:

```
$(window).height() + $(window).scrollTop() == $(document).height()
```

HTML 요소 스크롤:

```
$(element).height() + $(element).scrollTop() == $(element)[0].scrollHeight()
```

브라우저, 문서, 입력폼 이벤트(3)

예제 11-25

```
12: $(document).ready(function() {
13:     var no = 1;
14:   function getNextData() {
15:     var str = "";
16:     for (var i=0; i < 20; i++) {
17:        str += "<h2>Content : " + no + "</h2>";
18:        no++;
19:     }
20:     $("#panel").append(str);
21:   }
22:   getNextData();
23:
24:   $("#panel").on("scroll", function(e) {
25:     var p = $(this);
26:     var sh = p[0].scrollHeight - p.scrollTop();
27:     var ph = p.height();
28:
29:     if (sh == ph) {
30:        $("#processing").show().delay(500).fadeOut(50,function() {
31:           getNextData();
32:        });
33:     }
34:   });
```

```
36:   $("#gofirst").on("click", function() {
37:      $("#panel").scrollTop(0);
38:   });
39:
40:   $("#golast").on("click", function() {
41:      var sh = $("#panel")[0].scrollHeight;
42:      $("#panel").scrollTop(sh);
43:   });
44: });
```

```
48:   <button id="gofirst">처음으로</button>
49:   <button id="golast">마지막으로</button>
50:   <div id="container">
51:      <div id="panel">
52:      </div>
53:      <div id="processing" style="display:none; text-align: center;">
54:         <img id="imgPro" src="images/processing.gif" />
55:      </div>
56:   </div>
```

브라우저, 문서, 입력폼 이벤트(4)

- 예제 11-25 실행 결과

브라우저, 문서, 입력폼 이벤트(5)

focus, blur 이벤트

- 예제 11-26

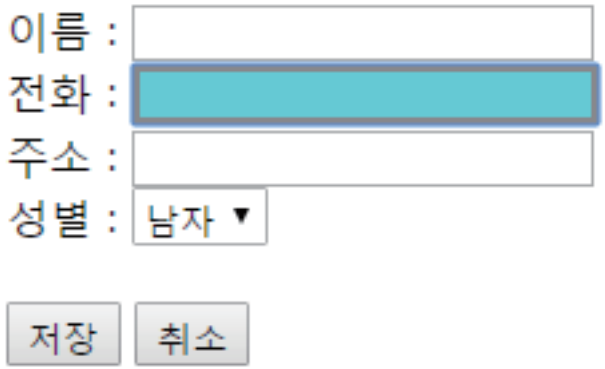
```
06: <style>
07: .focused { border: solid 3px gray; background-color: aqua; }
08: </style>
```

```
17:      <div id="container">
18:      이름 : <input id="name" type="text" class="test" /><br />
19:      전화 : <input id="tel" type="text" class="test" /><br />
20:      주소 : <input id="address" type="text" class="test" /><br />
21:      성별 : <select id="gender" name="gender" class="test">
22:         <option value="M">남자</option>
23:         <option value="F">여자</option>
24:      </select><br /><br />
25:      <button id="save" class="test">저장</button>
26:      <button id="cancel" class="test">취소</button>
27:   </div>
```

```
$("#container > .test").on("focus  blur", function(e) {
   $(this).toggleClass("focused");
});
```

브라우저, 문서, 입력폼 이벤트(6)

- 예제 11-26 실행 결과



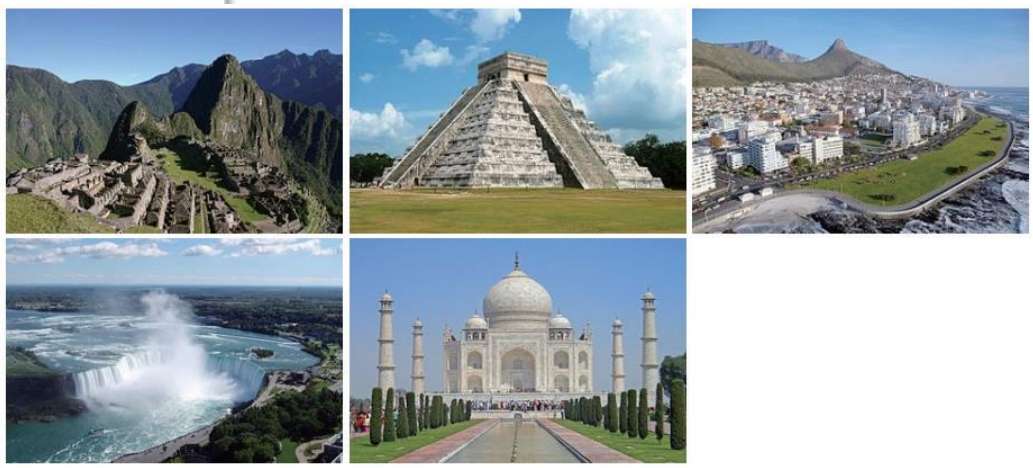
브라우저, 문서, 입력폼 이벤트(7)

error 이벤트

- 문서 요소의 완전한 로드에 실패했을 때 발생하는 이벤트
- 예제 11-27

```
10: <script type="text/javascript">
11: $(document).ready(function() {
12:   var images = ["machu.jpg", "chichenitza.jpg",
13:                 "capetown.jpg" ,"niagarafalls.jpg", "tajmahal.jpg" ];
14:
15:   $("img.travel").attr("src", function(i) {
16:      return "images/" + images[i];
17:   }).on("error", function() {
18:       $( this ).attr( "src", "images/noimage.jpg" );
19:   });
20:
21: });
22: </script>
23: </head>
24: <body>
25:   <img class="travel" />
26:   <img class="travel" />
27:   <img class="travel" />
28:   <img class="travel" />
29:   <img class="travel" />
30: </body>
```

정상적인 실행

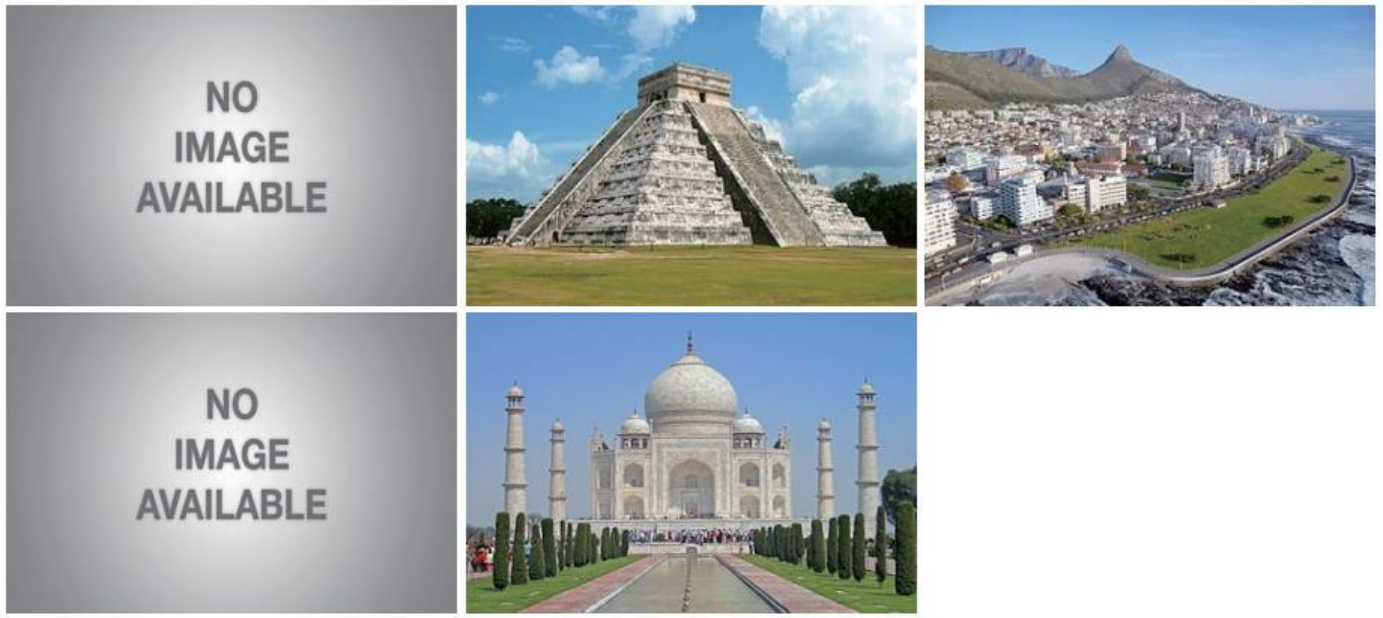


브라우저, 문서, 입력폼 이벤트(8)

- 예제 11-27 수정

```
var images = [ "this.jpg", "chichenitza.jpg",
      "capetown.jpg" ,"that.jpg", "tajmahal.jpg" ];
```

```
15:   $("img.travel").attr("src", function(i) {
16:      return "images/" + images[i];
17:   }).on("error", function() {
18:      $( this ).attr( "src", "images/noimage.jpg" );
19:   });
```



정리

- 이벤트 객체의 속성을 제대로 이해하고 활용하자
  - 동적인 UI를 작성할 때 유용하다.
  - 이벤트 객체를 통해서 제공되는 좌표, 키보드 관련 속성을 잘 이해하고 활용하자.
- 필요하다면 이벤트 위임을 활용하자.