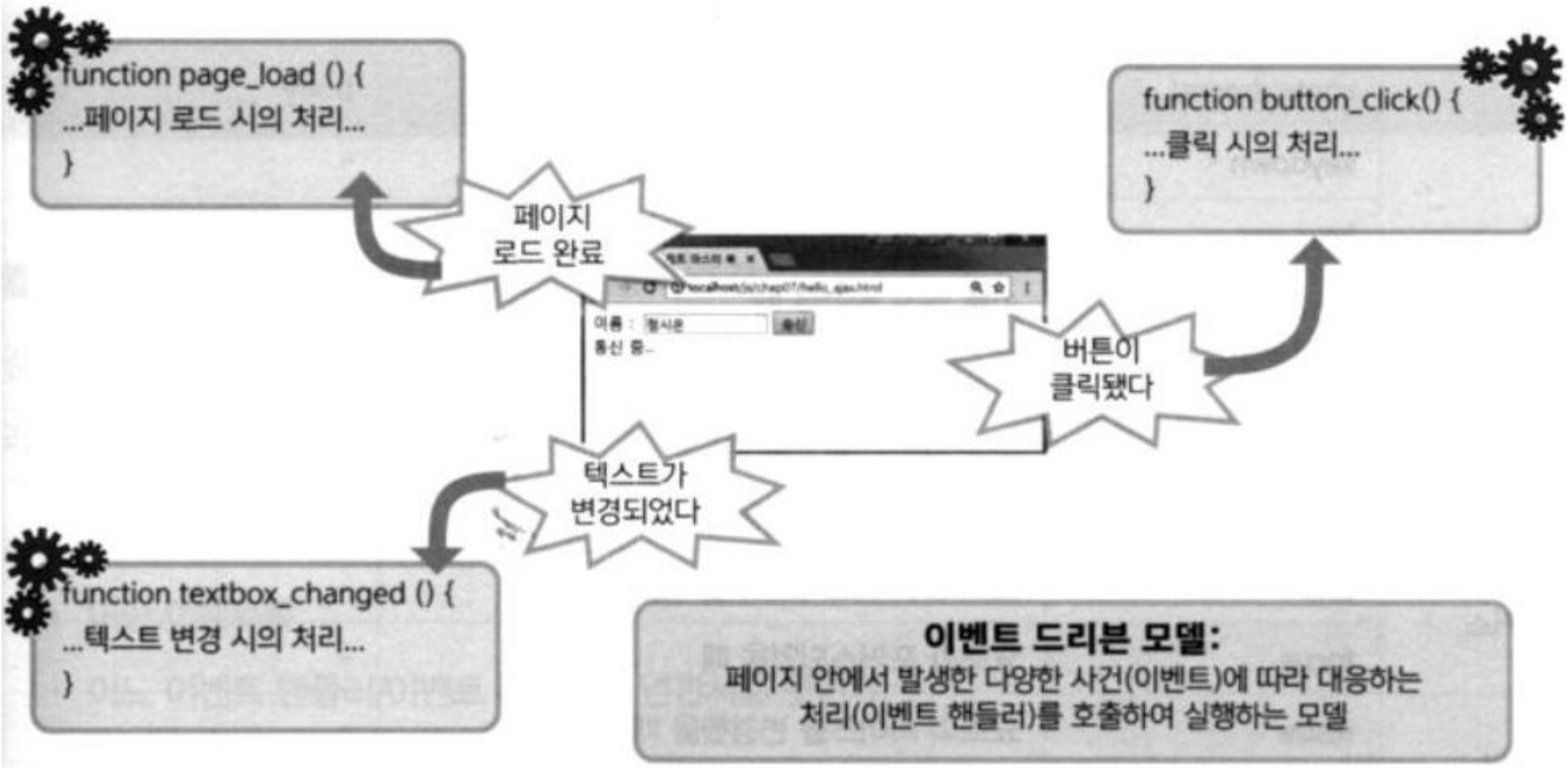
DOM

이벤트

❖ 이벤트를 트리커로 한 처리 실행하기 - 이벤트 구동 모델



❖ 이벤트

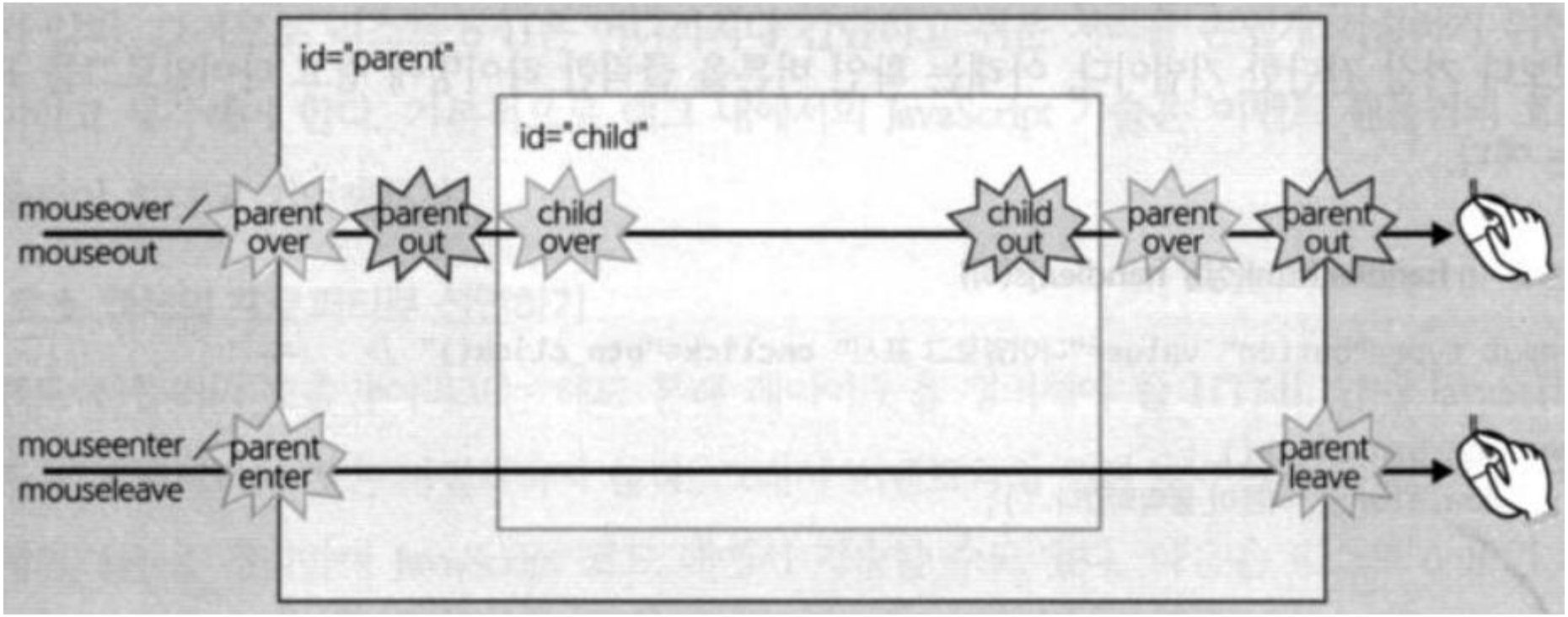
분류	이벤트명	발생 타이밍	주된 대상 요소
읽기	abort	이미지의 로딩이 중단되었을 때	img
	load	페이지, 이미지의 로딩 완료 시	body, img
	unload	다른 페이지로 이동할 때	body
마우스	click	마우스 클릭 시	-
	dblclick	더블 클릭 시	-
	mousedown	마우스 버튼을 눌렀을 때	-
	mouseup	마우스 버튼을 떼어 놓았을 때	-
	mousemove	마우스 포인터가 이동했을 때	-
	mouseover	요소에 마우스 포인터가 올라갔을 때	-
	mouseout	요소에서 마우스 포인터가 벗어났을 때	-
	mouseenter	요소에 마우스 포인터가 올라갔을 때	-
	mouseleave	요소에서 마우스 포인터가 벗어났을 때	-
	contextmenu	context menu가 표시되기 전	body

❖ 이벤트

키	keydown	키를 눌렀을 때	-
	keypress	키를 누른 상태	-
	keyup	키를 떼어 놓았을 때	-
폼	change	내용이 변경되었을 때	input(text), select
	reset	리셋 버튼이 눌렸을 때	form
	submit	서브밋 버튼이 눌렸을 때	form
포커스	blur	요소로부터 포커스가 벗어났을 때	-
	focus	요소가 포커스되었을 때	-
그 외	resize	요소의 사이즈를 변경했을 때	-
	scroll	스크롤했을 때	body

이벤트

❖ mouseover/mouseout과 mouseenter/mouseleave의 차이



이벤트

❖ 이벤트 리스너(핸들러) 정의하기

- 이벤트 처리시 고려사항
  - 어느 요소에서 발생했는가
  - 어떤 이벤트를
  - 어느 이벤트 핸들러/이벤트 리스너에 연관시킬 것인가

이벤트

❖ 이벤트 핸들러 연관 방법

- 태그 내의 속성으로 선언하기
- 요소 객체의 프로퍼티로 선언하기
- addEventListener 메서드를 사용하여 선언하기

이벤트

❖ 태그 내의 속성으로 선언하기

- <태그명 on태그명="자바스크립트 코드">

```
<input type="button" value="다이얼로그 표시" onclick="btn_click()" />
```

```
function btn_click() {
  window.alert('버튼이 클릭되었다.');
}
```

```
<input type="button" value="다이얼로그 표시"
  onclick="window.alert('버튼이 클릭되었다.');" />
```

이벤트

❖ 요소 객체의 프로퍼티로 선언하기

obj.on *event* = function () {statements}

obj : window 객체 또는 요소 객체

event : 이벤트명 *statements* : 이벤트가 발생할 때 수행할 처리

```
<input id="btn" type="button" value="다이얼로그 표시" />
```

```
window.onload = function() {
  document.getElementById('btn').onclick = function() {
    window.alert('버튼이 클릭되었다.');
  };
};
```

이벤트

❖ addEventListener 메서드로 선언하기

- 동일 이벤트에 여러개의 이벤트 핸들러 등록 가능

```
elem.addEventListener (type, listener, capture)
```

elem : 요소 객체 *type* : 이벤트의 종류

listener : 이벤트에 따라 수행할 작업 *capture* : 이벤트의 방향

```
<input id="btn" type="button" value="다이얼로그 표시" />
```

```
document.addEventListener('DOMContentLoaded', function() {
  document.getElementById('btn').addEventListener('click', function() {
    window.alert('버튼이 클릭되었다.');
  }, false);
}, false);
```

이벤트

❖ 초기화 이벤트

- onload
  - 콘텐츠 본체와 모든 이미지가 로드된 이후
- DOMContentLoaded
  - 콘텐츠 본체가 로드된 후(이미지의 로드를 기다리지 않음)

더 높은 수준의 이벤트 처리

❖ 이벤트 리스너(핸들러) 삭제하기

- elem.removeEventListener(type, listener, caputre)
  - elem: 요소 객체
  - type: 이벤트
  - listener: 삭제할 이벤트 리스너
  - capture: 이벤트의 전달 방향

더 높은 수준의 이벤트 처리

❖ 이벤트 리스너(핸들러) 삭제하기

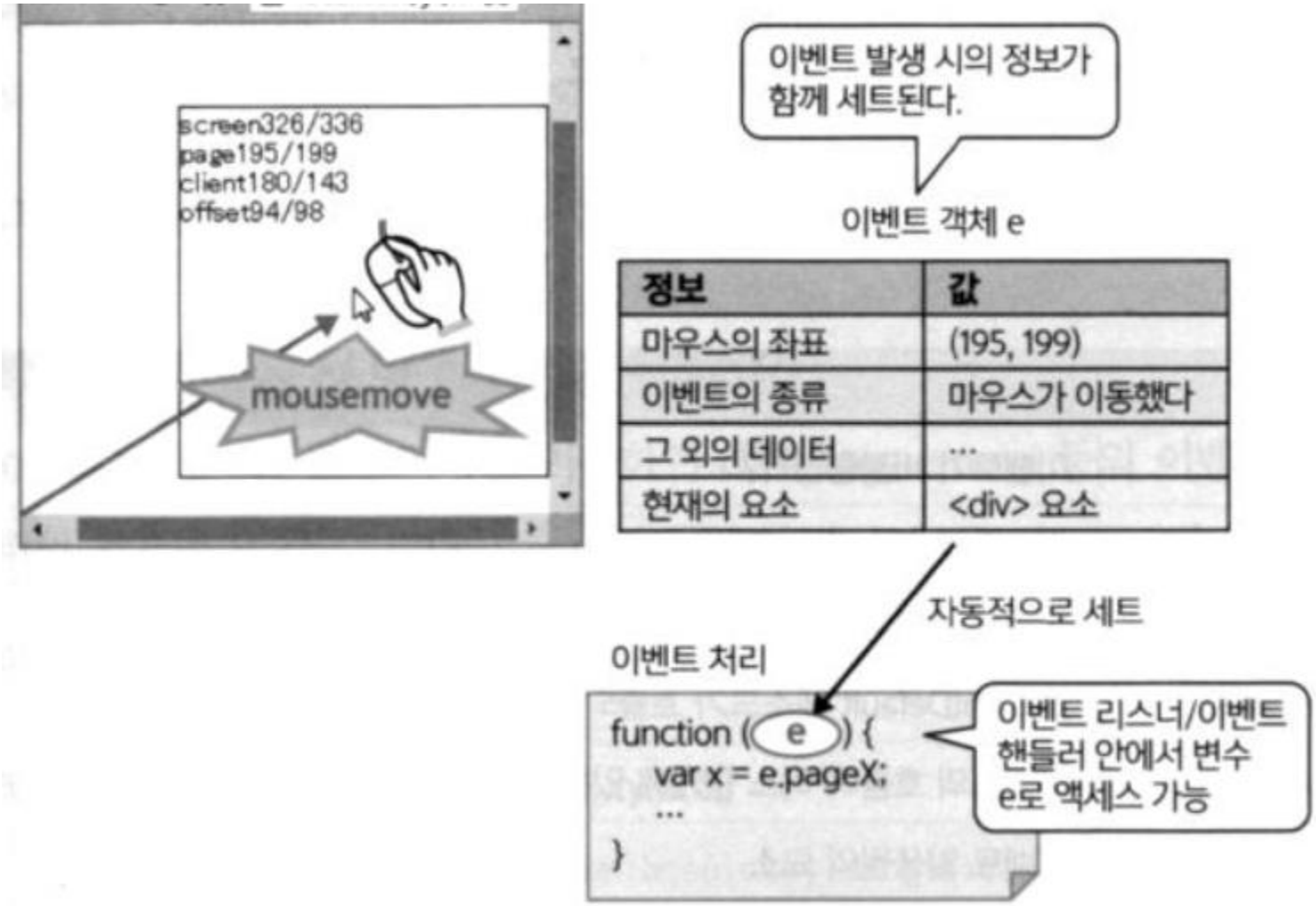
```
<form>
  <input id="btn" type="button" value="대화 상자 표시" />
</form>
```

```
document.addEventListener('DOMContentLoaded', function() {
  var btn = document.getElementById('btn');
  var listener = function() {
    window.alert('안녕하세요, 자바스크립트!');
  };

  btn.addEventListener('click', listener, false);

  btn.removeEventListener('click', listener, false);
}, false);
```

더 높은 수준의 이벤트 처리

❖ 이벤트 객체

- 발생한 이벤트와 관련된 추가 정보를 가지는 객체



더 높은 수준의 이벤트 처리

❖ 이벤트 객체

분류	멤버	개요
일반	bubbles	이벤트가 버블링인가?
	cancelable	이벤트가 취소 가능한가?
	currentTarget	이벤트 버블에서 현재 요소를 취득
	defaultPrevented	preventDefault 메소드가 호출되었는가?
	eventPhase	이벤트의 흐름이 어느 단계에 있는가?
	target	이벤트 발생원의 요소
	type	이벤트의 종류(click, mouseover 등)
	timeStamp	이벤트의 작성 일시를 취득

더 높은 수준의 이벤트 처리

❖ 이벤트 객체

좌표	clientX	이벤트의 발생 좌표(브라우저상에서의 X좌표)
	clientY	이벤트의 발생 좌표(브라우저상에서의 Y좌표)
	screenX	이벤트의 발생 좌표(스크린상에서의 X좌표)
	screenY	이벤트의 발생 좌표(스크린상에서의 Y좌표)
	pageX	이벤트의 발생 좌표(페이지상의 X좌표)
	pageY	이벤트의 발생 좌표(페이지상의 Y좌표)
	offsetX	이벤트의 발생 좌표(요소상의 X좌표)
	offsetY	이벤트의 발생 좌표(요소상의 Y좌표)

더 높은 수준의 이벤트 처리

❖ 이벤트 객체

<table>
<tr><td rowspan="7">키보드/마우스</td><td>button</td><td>마우스의 어느 버튼이 눌리고 있는가?
<table>
<tr><th>버튼의 종류</th><th>반환값</th></tr>
<tr><td>좌측 버튼</td><td>0</td></tr>
<tr><td>우측 버튼</td><td>2</td></tr>
<tr><td>가운데 버튼</td><td>1</td></tr>
</table></td></tr>
<tr><td>key</td><td>눌린 키의 값</td></tr>
<tr><td>keyCode</td><td>눌린 키의 코드</td></tr>
<tr><td>altKey</td><td>alt 키가 눌린 상태인가</td></tr>
<tr><td>ctrlKey</td><td>ctrl 키가 눌린 상태인가</td></tr>
<tr><td>shiftKey</td><td>shift 키가 눌린 상태인가</td></tr>
<tr><td>metaKey</td><td>meta 키가 눌린 상태인가</td></tr>
</table>

더 높은 수준의 이벤트 처리

❖ 이벤트 객체

```
<form>
  <input id="btn" type="button" value="클릭" />
</form>
```

```
document.addEventListener('DOMContentLoaded', function() {
  document.getElementById('btn').addEventListener('click', function(e) {
    var target = e.target;
    console.log('발생원:' + target.nodeName + '/' + target.id);
    console.log('종류:' + e.type);
  }, false);
}, false);
```

더 높은 수준의 이벤트 처리

❖ 이벤트 발생 시 마우스 정보 취득

```
<div id="main" style="position:absolute; margin:50px;
 top:50px; left:50px; width:200px; height:200px;
 border:1px solid Black"></div>
```

```
document.addEventListener('DOMContentLoaded', function() {
  var main = document.getElementById('main');
  main.addEventListener('mousemove', function(e) {
    main.innerHTML = 'screen' + e.screenX + '/' + e.screenY + '<br />'
      + 'page' + e.pageX + '/' + e.pageY + '<br />'
      + 'client' + e.clientX + '/' + e.clientY + '<br />'
      + 'offset' + e.offsetX + '/' + e.offsetY + '<br />';
  }, false);
}, false);
```

프로퍼티	개요
screenX/screenY	스크린상의 좌표
pageX/pageY	페이지상의 좌표
clientX/clientY	표시 영역상의 좌표
offsetX/offset	요소 영역상의 좌표

더 높은 수준의 이벤트 처리

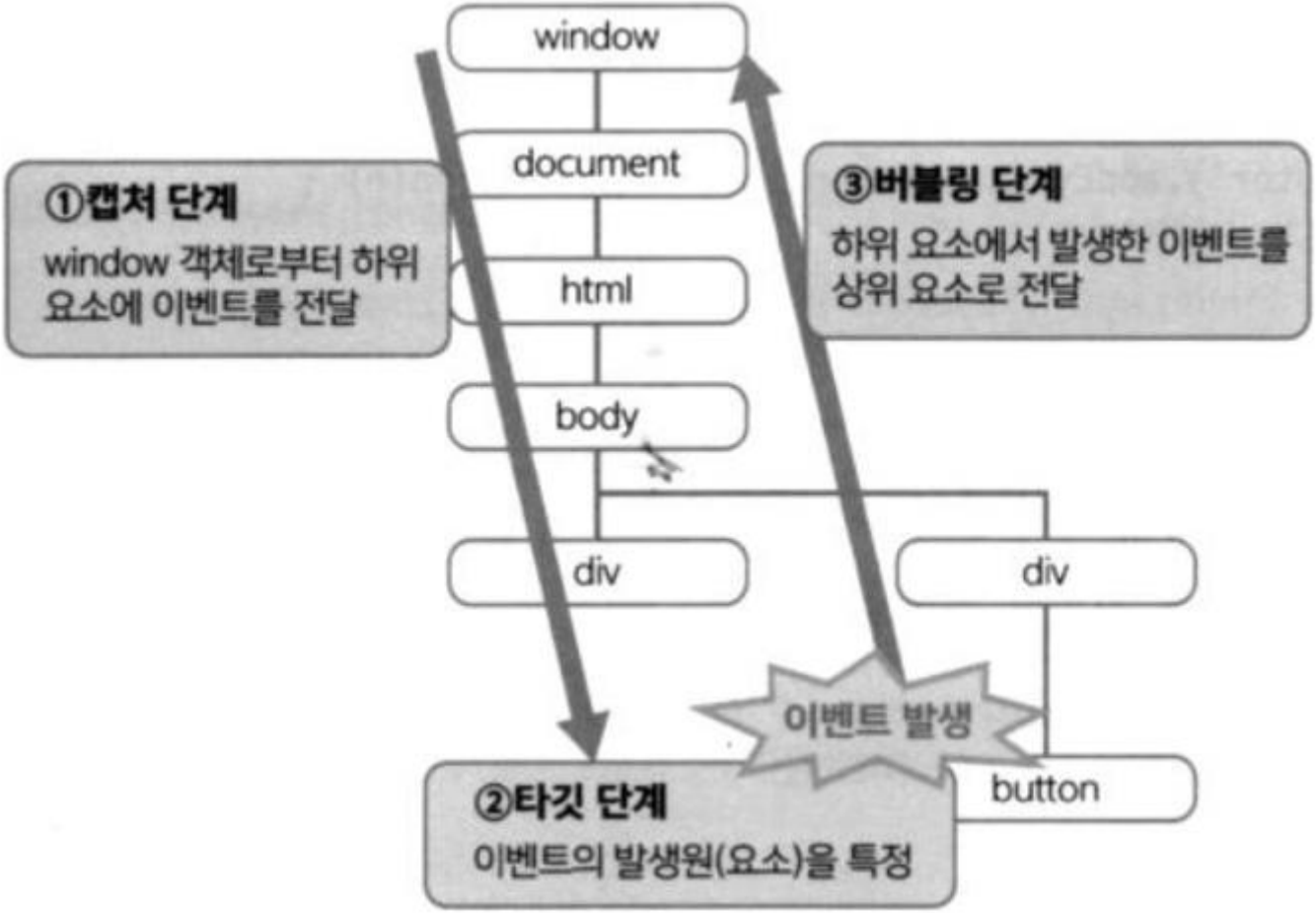
❖ 이벤트 발생시 키 정보 취득

```
<form>
  <label for="key">키 입력:</label>
  <input type="text" id="key" size="10" />
</form>
```

```
document.addEventListener('DOMContentLoaded', function() {
  document.getElementById('key').addEventListener('keydown', function(e) {
    console.log('키 코드:' + e.keyCode);
  }, false);
}, false);
```

더 높은 수준의 이벤트 처리

❖ 이벤트의 전달



더 높은 수준의 이벤트 처리

❖ 이벤트의 전달

```
<div id="outer">
  <p>outer요소</p>
  <a id="inner" href="http://www.wings.msn.to">inner요소</a>
</div>
```

```
document.addEventListener('DOMContentLoaded', function() {
  document.getElementById('inner').addEventListener('click', function(e) {
    window.alert('#inner리스너가 발생하였다.');
  }, false);

  document.getElementById('inner').addEventListener('click', function(e) {
    window.alert('#inner리스너2가 발생하였다.');
  }, false);

  document.getElementById('outer').addEventListener('click', function(e) {
    window.alert('#outer리스너가 발생하였다.');
  }, false);
}, false);
```

더 높은 수준의 이벤트 처리

❖ 이벤트의 전달 취소

메소드	전달	다른 리스너	디폴트 동작
stopPropagation	정지	-	-
stopImmediatePropagation	정지	정지	-
preventDefault	-	-	정지

더 높은 수준의 이벤트 처리

❖ 이벤트의 전달 취소

```
<div id="outer">
  <p>outer요소</p>
  <a id="inner" href="http://www.wings.msn.to">inner요소</a>
</div>
```

```
document.addEventListener('DOMContentLoaded', function() {
  document.getElementById('inner').addEventListener('click', function(e) {
    window.alert('#inner리스너가 발생하였다.');
    e.stopPropagation();
    //e.stopImmediatePropagation();
    //e.preventDefault();
  }, false);

  document.getElementById('inner').addEventListener('click', function(e) {
    window.alert('#inner리스너2가 발생하였다.');
  }, false);

  document.getElementById('outer').addEventListener('click', function(e) {
    window.alert('#outer리스너가 발생하였다.');
  }, false);
}, false);
```

더 높은 수준의 이벤트 처리

❖ 이벤트 리스너(핸들러)에서의 this

- 이벤트 발생원이 this가 됨

```
<input id="btn" type="button" value="클릭" />
```

```
document.addEventListener('DOMContentLoaded', function() {
  var data = {
    title: 'Java포켓 레퍼런스',
    price: 26800,
    show: function() {
      console.log(this.title + '/' + this.price + '원');
    }
  };

  document.getElementById('btn').addEventListener(
    'click', data.show, false);

  // document.getElementById('btn').addEventListener(
  //  'click', data.show.bind(data), false);
}, false);
```

더 높은 수준의 이벤트 처리

❖ 화살표 함수의 this

- 함수가 포함된 객체가 this가 됨

```
<input id="btn" type="button" value="클릭" />
```

```
document.addEventListener('DOMContentLoaded', function() {
  var Counter = function(elem) {
    this.count = 0;
    this.elem = elem;

    elem.addEventListener('click', () => {
      this.count++;
      this.show();
    }, false);
  };

  Counter.prototype.show = function() {
    console.log(this.elem.id + '는' + this.count + '회 클릭되었다.');
  }

  var c = new Counter(document.getElementById('btn'));
}, false);
```