기본 효과 메서드(1)

기본 효과 메서드 10개

[show – hide – toggle]

[slideDown – slideUp – slideToggle]

[fadeIn – fadeOut – fadeToggle – fadeTo]

[**표 12-01** : 기본 효과 메서드]

메서드	설명
show( )	선택된 요소를 나타나게 한다.
hide( )	선택된 요소를 사라지게 한다.
toggle( )	선택된 요소를 나타나게 하거나 사라지게 한다.
slideDown( )	선택된 요소를 슬라이딩 모션으로 나타나게 한다.
slideUp( )	선택된 요소를 슬라이딩 모션으로 사라지게 한다.
slideToggle( )	선택된 요소를 슬라이딩 모션으로 나타나게 하거나 사라지게 한다.
fadeIn()	선택된 요소의 불투명도를 변화시켜 나타나게 한다.
fadeOut( )	선택된 요소의 불투명도(opacity)를 변화시켜 사라지게 한다.
fadeToggle( )	선택된 요소의 불투명도(opacity)를 변화시켜 나타나게 하거나 사라지게 한다.
fadeTo( )	선택된 요소를 지정한 불투명도(opacity:0~1)로 서서히 변경시킨다.

기본 효과 메서드(2)

기본 효과 메서드 사용법

- 메서드들 모두 사용법 동일

- show()
- show([duration] [, complete])
- show(duration[, easing] [, complete])
- show(options)

duration 효과 지속 시간

complete 효과의 실행이 완료되면 실행할 콜백 함수

easing 이징 효과. 기본값은 swing임. linear로 설정 가능함. 더욱 다양한 이징 효과를 사용하려면 jQuery UI 구성 요소를 설치해야 함.

options 객체로서 다음 속성을 지정할 수 있다.

duration 애니메이션의 실행 지속 시간을 지정함(기본값:400)

easing 이징 효과

queue 애니메이션을 효과 큐(effect queue)에 넣을지(true/false) 또는 큐 이름(string)을 지정함

start 효과를 시작할 때 호출되는 함수

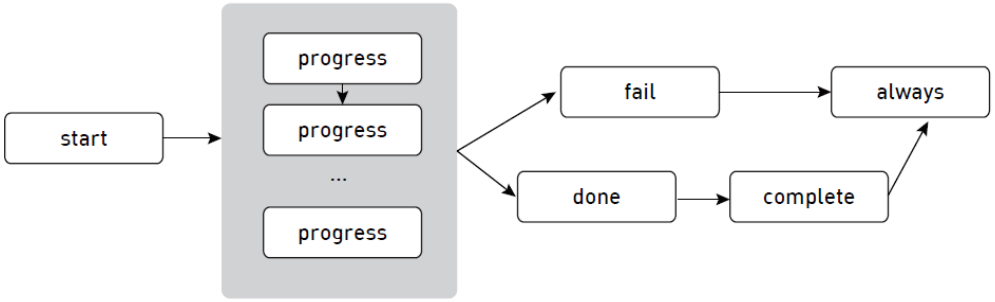
progress 효과가 진행되는 동안 진행 상태를 파악할 수 있도록 호출되는 함수

done 효과를 완료할 때 호출되는 함수

complete 정상적으로 효과가 완료되고 나면 호출되는 함수

fail 효과 실행이 완료되지 못했을 때 호출되는 함수

always 효과 실행의 완료 유무와 관계없이 마지막에 호출되는 함수



기본 효과 메서드(3)

예제 12-01

```
22:   <button id="show">show</button>
23:   <button id="hide">hide</button>
24:   <button id="toggle">toggle</button>
25:   <br />
26:   <img id="a" src="images/free3.jpg" style="display:none" />
```

```
07: <script type="text/javascript">
08:   $(document).ready(function() {
09:      $("#show").click(function() {
10:         $("#a").show(1000);
11:      });
12:      $("#hide").click(function() {
13:         $("#a").hide(1000);
14:      });
15:      $("#toggle").click(function() {
16:         $("#a").toggle(1000);
17:      });
18:   });
19: </script>
```

기본 효과 메서드(4)

예제 12-02

- 다양한 옵션 사용 예

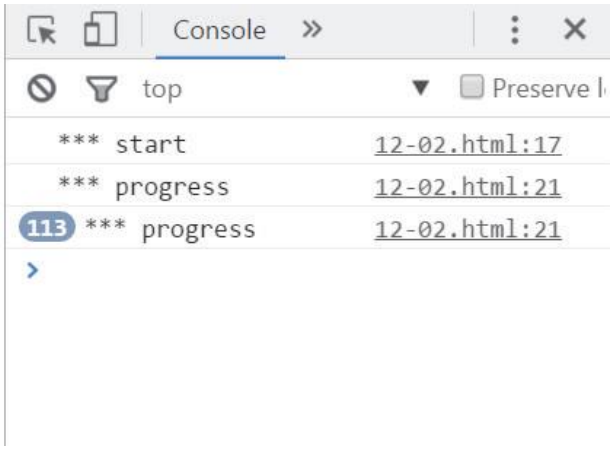
```
12:  $("#show").click(function() {
13:     $("#a").show({
14:        duration: 3000,
15:        easing:"linear",
16:        start : function(promise) {
17:           console.log("*** start");
18:        },
19:        progress : function(promise, prog, remain) {
20:           var percent = Math.round(prog * 100) + "%";
21:           console.log("*** progress");
22:           $("#progress").html("진행률 : " + percent +
23:                 ", 남은 시간 : " + remain + "ms");
24:        },
25:        done : function() {
26:           console.log("*** done : ");
27:        },
28:        complete : function() {
29:           console.log("*** complete" );
30:        },
31:        fail : function() {
32:           console.log("*** fail : ");
```

```
33:        },
34:        always : function() {
35:           console.log("*** always : ");
36:        }
37:     });
38:  });
```

```
43:  <button id="show">show</button><br />
44:  <span id="progress"></span>
45:  <br />
46:  <img id="a" src="images/free2.jpg" />
```

기본 효과 메서드(5)

- 예제 12-02 실행 결과



- jQuery 효과 메서드는 Deferred 객체를 이용한 비동기 처리 방식임

효과의 처리 방식(1)

- jQuery 효과의 처리 방식은 비동기 방식
  - 여러 개 요소에 효과를 적용하면 동시에 실행된다.
  - 예제 12-03

```
24:   <button id="toggle_img">toggle</button>
25:   <div id="container">
26:      <img id="a1" class="pic" src="images/free1.jpg" />
27:      <img id="a2" class="pic" src="images/free2.jpg" />
28:      <img id="a3" class="pic" src="images/free3.jpg" />
29:      <img id="a4" class="pic" src="images/free4.jpg" />
30:      <img id="a5" class="pic" src="images/free5.jpg" />
31:   </div>
```

```
13:   $("#toggle_img").click(function() {
14:      $("#a1").toggle(1000);
15:      $("#a2").toggle(1000);
16:      $("#a3").toggle(1000);
17:      $("#a4").toggle(1000);
18:      $("#a5").toggle(1000);
19:   });
```

효과의 처리 방식(2)

- 예제 12-03 실행 결과
  - 14~18행까지 순차적으로 toggle() 메서드를 호출했지만 동시에 효과가 일어남
  - 비동기적으로 실행하기 때문이다.

  - 만일 순차적으로 실행하고 싶다면 complete 콜백함수를 이용해야 한다
  - 14~18행 코드는 $("img.pic").toggle(1000);처럼 한 줄로 표현할 수 있지만 순차적으로 호출한다는 의미를 강하게 부여하기 위해 하나씩 호출하도록 변경했다.

효과의 처리 방식(3)

- 예제 12-04
  - 순차적으로 실행하기 위해 complete 함수 이용
  - 재귀 함수를 사용해 다음 요소를 찾아 효과 적용함.

```
11: <script type="text/javascript">
12: $(document).ready(function() {
13:   function toggleSeq(elem) {
14:      elem.toggle(1000, function() {
15:         var next = elem.next();
16:         if (next.length > 0) {
17:            toggleSeq(next);
18:         }
19:      });
20:   }
21:
22:   $("#toggle_img").click(function() {
23:      toggleSeq($("img.pic").first());
24:   });
25: });
26: </script>
```

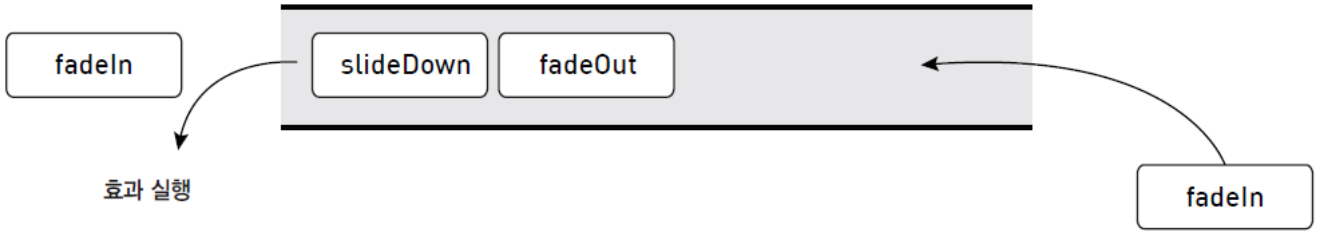
효과의 처리 방식(4)

동일한 요소에 여러 개 효과를 적용하면?

- Effect Queue(Animation Queue)를 이용함. Queue에 적재된 순서대로 실행함.
- 예제 12-05

```
22:   <button id="start">Start</button>
23:   <br />
24:   <img id="a" src="images/free1.jpg" />
```

```
12:   $("#start").click(function() {
13:      $("#a").slideUp(2000);
14:      $("#a").slideDown(2000);
15:      $("#a").fadeOut(2000);
16:      $("#a").fadeIn(2000);
17:   });
```



animate( ) 메서드와 지연 메서드(1)

기본 효과 메서드에 비해 다양한 이벤트 처리 가능

- 기본 효과 메서드는 어차피 내부적으로 animate() 메서드 사용

```
> $.prototype.slideDown
< function ( speed, easing, callback ) {
        return this.animate( props, speed, easing, callback );
    }
```

- animate() 사용 방법

```
animate( properties [, duration ] [, easing ] [, complete ] )

animate( properties, options )
```

  - 기본 효과 메서드와의 차이점은 첫번째 인자가 하나 더 있다는 점(properties)

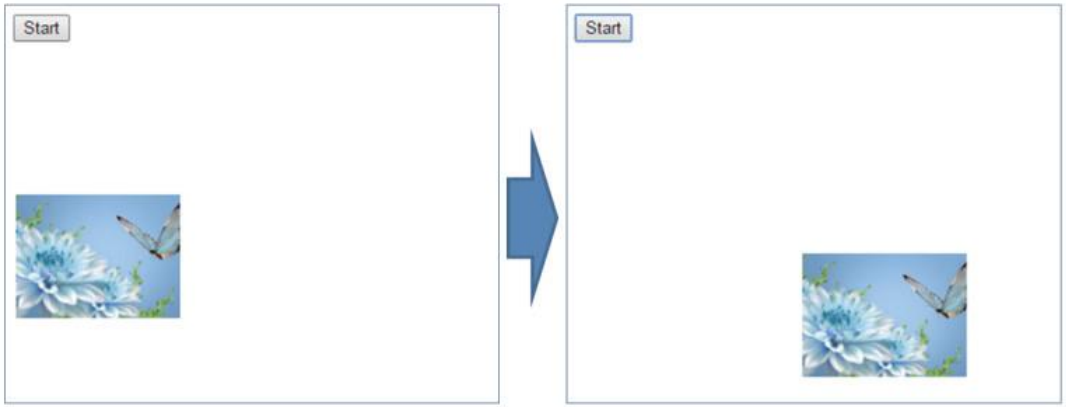
animate( ) 메서드와 지연 메서드(2)

예제 12-06

- animate() 메서드로 animation queue 사용
  - 상대적 좌표 사용 : +=, -= 표기법 사용

```
29:   <button id="start">Start</button>
30:   <br />
31:   <img id="a" src="images/free5.jpg" />
```

```
12:   $("#start").click(function() {
13:      $("#a").animate({
14:         left : 200,
15:         top : 200
16:      }, 2000).animate({
17:         width : "+=128",
18:         height: "+=96",
19:         top: "-=48",
20:         left: "-=64"
21:      }, 2000).animate({
22:         opacity: 0.3
23:      }, 1000);
24:   });
```



효과 정지와 효과 큐 관리(1)

효과 정지와 효과 큐 관리 메서드

[**표 12-02** : 효과 정지 메서드와 효과 큐 관리 메서드]

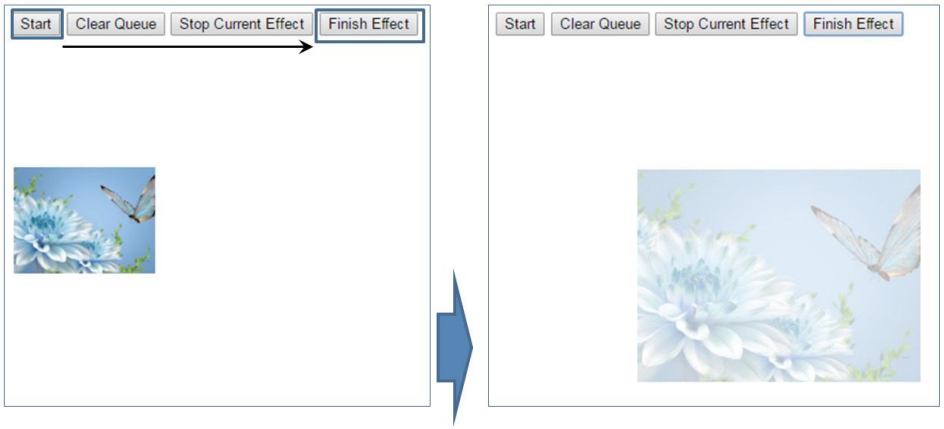
메서드	설명
clearQueue( )	선택된 요소에 대해 효과 큐에 적재된 효과를 삭제한다. 현재 실행 중인 효과는 정상적으로 실행된다.
dequeue( )	선택된 요소에 대한 효과 큐에서 다음 함수를 꺼내 실행한다.
queue( )	선택된 요소에 대한 효과 큐를 리턴한다.
stop( )	현재 실행 중인 효과를 정지시킨다. 효과 큐에 적재된 효과가 있다면 다음 효과를 실행하게 된다.
finish( )	현재 실행 중인 효과를 정지시키고, 큐에 적재된 효과도 제거한 후 요소를 효과 실행이 완료된 상태로 즉시 변경한다.

효과 정지와 효과 큐 관리(2)

예제 12-07

```
12:  $("#start").click(function() {
13:     $("#a").animate({
14:        left : 200,
15:        top : 200
16:     }, 2000).animate({
17:        width : "+=128",
18:        height: "+=96",
19:        top: "-=48",
20:        left: "-=64"
21:     }, 2000).delay(4000).animate({
22:        opacity:0.3
23:     }, 1000);
24:  });
25:
26:  $("#clearq").click(function() {
27:     $("#a").clearQueue();
28:  });
29:
30:  $("#stop").click(function() {
31:     $("#a").stop();
32:  });
33:
34:  $("#finish").click(function() {
35:     $("#a").finish();
36:  });
```

```
41:  <button id="start">Start</button>
42:  <button id="clearq">Clear Queue</button>
43:  <button id="stop">Stop Current Effect</button>
44:  <button id="finish">Finish Effect</button>
45:  <br />
46:  <img id="a" src="images/free5.jpg" />
```



요소 상태를 완료 상태로 즉시 변경

효과 정지와 효과 큐 관리(3)

- clearQueue()
  - 효과 큐에 적재된 함수 삭제. 현재 실행중인 함수는 삭제하지 않음.
- stop()

```
stop( clearQueue, jumpToEnd )
```

  - 현재 실행중인 효과를 중지시키지만 첫번째 인자를 true로 지정하면 효과 큐도 삭제함.
  - jumpToEnd 파라미터는 요소를 마지막 상태로 변경시킴
- finish()
  - 모든 효과를 종료하고 선택된 요소를 효과가 완료된 상태로 즉시 변경한다.

효과 정지와 효과 큐 관리(4)

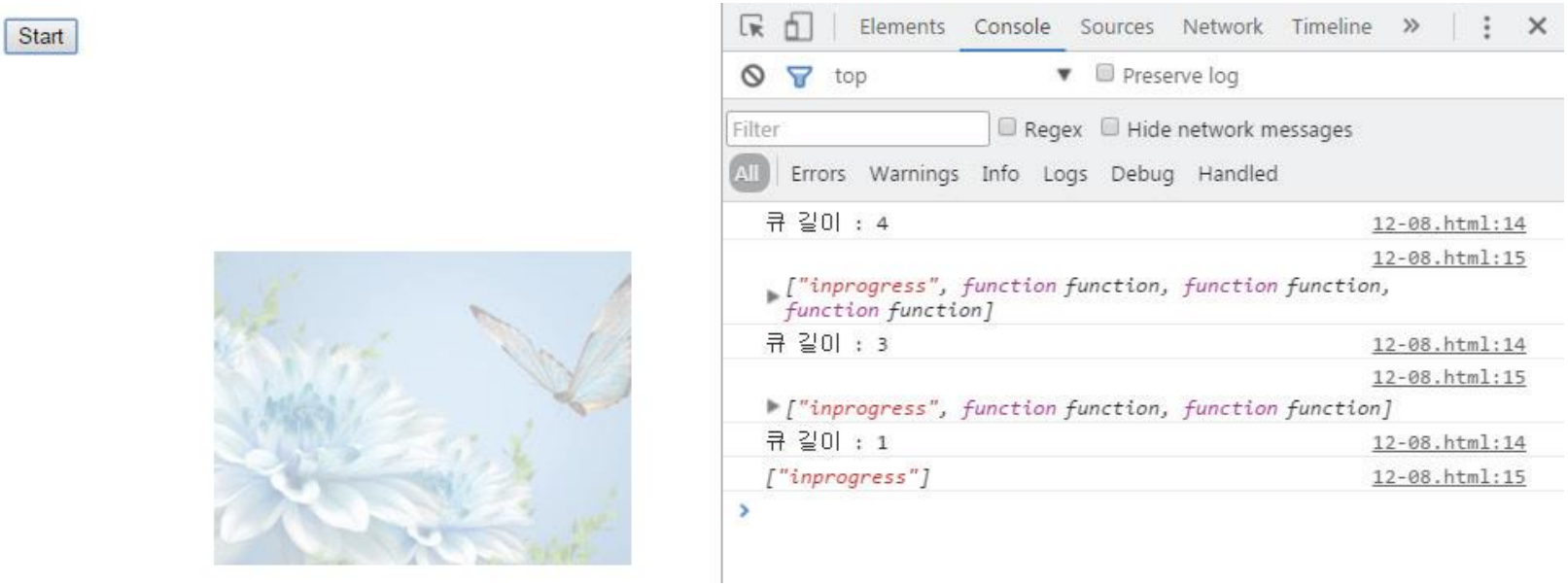
큐 정보 관리

- complete 콜백으로 completeCB 함수 호출
- 선택된 요소의 큐 정보를 얻기 위해 .queue() 메서드 호출
- 예제 12-08

```
12: function completeCB() {
13:   var q = $("#a").queue().slice();
14:   console.log("큐 길이 : " + q.length)
15:   console.log(q);
16: }
17:
18: $("#start").click(function() {
19:   $("#a").animate({
20:     left : 200,
21:     top : 200
22:   }, 2000, completeCB).animate({
23:     width : "+=128",
24:     height: "+=96",
25:     top: "-=48",
26:     left: "-=64"
27:   }, 2000, completeCB).delay(4000).animate({
28:     opacity:0.3
29:   }, 1000, completeCB);
30: });
```

효과 정지와 효과 큐 관리(5)

- 예제 12-08 실행 결과



전역 효과 설정

- 저사양 PC이거나 효과가 필요없는 경우
  - 효과가 작동하지 않도록 설정할 수 있음
  - 효과의 적용 주기를 더 길게 설정할 수 있음
  - $.fx.off
    - 이 값을 true로 지정하면 애니메이션 효과 off
  - $.fx.interval
    - 이 값을 ms 단위로 지정. 애니메이션 효과 적용 간격
    - 이 값이 작을 수록 애니메이션 효과가 더 부드럽게 작동함.

jQuery UI의 이징과 효과(1)

- jQuery UI는 다양한 이징과 추가적인 효과를 제공한다.
- jQuery UI의 설치
  - 직접 내려받아 사용할수도 있지만 CDN도 제공함.
  - 예제 12-10 : 이징

```
11: <script src="https://code.jquery.com/jquery-3.1.0.js"></script>
12: <script src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"></script>
13: <script type="text/javascript">
14: $(document).ready(function() {
15:   for (var k in $.easing) {
16:      $("<button></button>").html(k).appendTo("#buttonlist");
17:   }
18:
19:   $("#buttonlist > button").click(function() {
20:      var easing = $(this).html();
21:      $("#tiger").animate({ left :"+=250" }, 1500, easing);
22:      $("#tiger").delay(500);
23:      $("#tiger").animate({ left : "-=250" }, 1500, easing);
24:   });
25: });
26: </script>
```

jQuery UI의 이징과 효과(2)

- 예제 12-10 (이어서)

```
29:   <div id="buttonlist">
30:   </div>
31:   <div id="container">
32:      <img id="tiger" src="images/tiger.jpg" />
33:   </div>
```

linear | swing | easeInQuad | easeOutQuad | easeInOutQuad | easeInCubic | easeOutCubic
easeInOutCubic | easeInQuart | easeOutQuart | easeInOutQuart | easeInQuint | easeOutQuint
easeInOutQuint | easeInExpo | easeOutExpo | easeInOutExpo | easeInSine | easeOutSine
easeInOutSine | easeInCirc | easeOutCirc | easeInOutCirc | easeInElastic | easeOutElastic
easeInOutElastic | easeInBack | easeOutBack | easeInOutBack | easeInBounce | easeOutBounce
easeInOutBounce

jQuery UI의 이징과 효과(3)

- 예제 12-11 : 효과

```
13: <script src="https://code.jquery.com/jquery-3.1.0.js"></script>
14: <script src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"></script>
15: <script type="text/javascript">
16: $(document).ready(function() {
17:   for (var k in $.effects.effect) {
18:      $("<button></button>").html(k).appendTo("#buttonlist");
19:   }
20:
21:   $("#buttonlist > button").click(function() {
22:      var eff = $(this).html();
23:
24:      var options = {};
25:      if ( eff === "scale" ) {
26:        options = { percent: 10 };
27:      } else if ( eff === "transfer" ) {
28:        options = { to: "button:last", className: "trans_effect" };
29:      } else if ( eff === "size" ) {
30:         options = { to: { width: 300, height: 30 } };
31:      }
32:
33:      $("#box").effect(eff, options, 1000).delay(1000).hide(1, function() {
34:         $(this).removeAttr("style");
35:      });
36:   });
37: });
38: </script>
```

jQuery UI의 이징과 효과(4)

새로운 이미지 슬라이드 쇼

- 예제 12-12 : HTML 마크업 작성

```
06: <script src="https://code.jquery.com/jquery-3.1.0.js"></script>
07: <script src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"></script>
08: <script type="text/javascript">
09: $(document).ready(function() {
10:
11: });
12: </script>
13: </head>
14: <body>
15:      <div id="page1" class="container">
16:      <h2>이미지 1</h2>
17:      <img src="images/free1.jpg" />
18:   </div>
19:   <div id="page2" class="container">
20:      <h2>이미지 2</h2>
21:      <img src="images/free2.jpg" />
22:   </div>
```

```
23:   <div id="page3" class="container">
24:      <h2>이미지 3</h2>
25:      <img src="images/free3.jpg" />
26:   </div>
27:   <div id="page4" class="container">
28:      <h2>이미지 4</h2>
29:      <img src="images/free4.jpg" />
30:   </div>
31:   <div id="page5" class="container">
32:      <h2>이미지 5</h2>
33:      <img src="images/free5.jpg" />
34:   </div>
```

jQuery UI의 이징과 효과(5)

- 예제 12-13 : 스타일

[예제 12-13]

```
<style>
.container { margin: 5px 5px 5px 5px; padding:15px 15px 15px 15px;
     background-color: #F7BE81; border: solid 1px gray;
     position:absolute ; top:8px; left:8px;
     width:640px; height:600px;
     box-shadow: 0 0 10px #000000;}
.left { position:absolute; bottom: 5px; left: 5px; }
.right { position:absolute; bottom: 5px; right: 5px;}
</style>
```

- 예제 12-14 : 기능 수행 코드

[예제 12-14]

```
    $("div.container").each(function() {
        var btns = "<button class='left'>&lt;</button>" +
                   "<button class='right'>&gt;</button>";
        $(this).append(btns);
    });

    var currentView = $(".container").hide().first().show();
```

jQuery UI의 이징과 효과(6)

- 예제 12-14 : 이어서

```
08:
09:     function moveLeft() {
10:         var prev = currentView.prev();
11:         if (prev.length > 0) {
12:             prev.show("slide", { direction: "left" }, 500);
13:             currentView.hide("slide", { direction: "right" }, 500);
14:             currentView = prev;
15:         }
16:     }
17:
18:     function moveRight() {
19:         var next = currentView.next();
20:         if (next.length > 0) {
21:             next.show("slide", { direction: "right" }, 500);
22:             currentView.hide("slide", { direction: "left" }, 500);
23:             currentView = next;
24:         }
25:     }
```

```
26:
27:     $(".left").click(function() {
28:         moveLeft();
29:     });
30:
31:     $(".right").click(function() {
32:         moveRight();
33:     });
34:
35:     //커서 키 처리
36:     $(document).keydown(function(e) {
37:         //e.keyCode : <- 37    -> 39
38:         var view;
39:         if (e.keyCode == 37) {
40:             moveLeft();
41:         }
42:
43:         if (e.keyCode == 39) {
44:             moveRight();
45:         }
46:     });
```

jQuery UI의 이징과 효과(7)

- 예제 12-14 실행 결과