

이벤트 기초 및 활용



소프트웨어학부
진혜진



목차

1. 이벤트
2. 이벤트 객체
3. 이벤트 흐름
4. 마우스 핸들링
5. 문서와 이미지 로딩, onload
6. 폼과 이벤트 활용

1. 이벤트

3

□ 이벤트

- 마우스 클릭, 키보드 입력, 이미지나 HTML 문서의 로딩, 타이머의 타임아웃 등 사용자의 입력 행위, 문서나 브라우저의 상태 변화를 자바스크립트 코드에게 알리는 통지(notification)

□ 이벤트 리스너

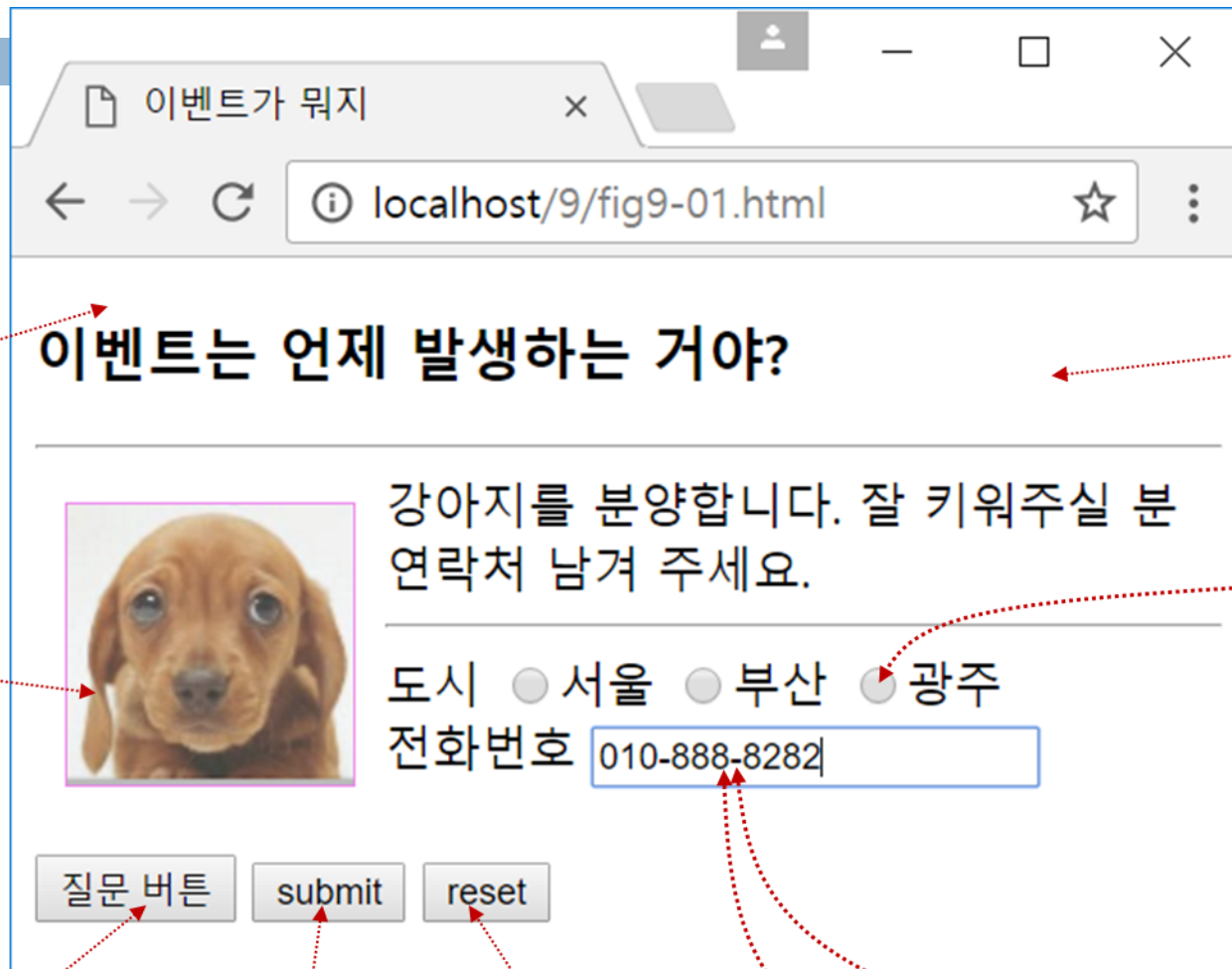
- 발생한 이벤트에 대처하기 위해 작성된 자바스크립트 코드

□ 이벤트 종류

- 이벤트 리스너 이름은 이벤트 이름 앞에 on을 덧붙임
- 예) onmousedown : mousedown 이벤트의 리스너
onkeydown : keydown 이벤트의 리스너

1. 이벤트

4



load 이벤트
(HTML 문서 전체 로딩 완료 시)

dblclick 이벤트
(마우스 더블클릭 시)

load 이벤트
(이미지의 로딩 완료 시)

change 이벤트
(라디오버튼 선택 시)

click 이벤트
(마우스 클릭 시)

submit 이벤트
(submit 버튼 클릭 시)

reset 이벤트
(reset 버튼 클릭 시)

keypress 이벤트
(키를 누를 때)

keyup 이벤트
(누른 키를 놓을 때)

resize 이벤트
(윈도우 크기 변경 시)

1. 이벤트

5

- 이벤트 리스너 만들기
 - ▣ HTML 태그 내에 작성
 - ▣ DOM 객체의 이벤트 리스너 프로퍼티에 작성
 - ▣ DOM 객체의 `addEventListener()` 메소드 이용

- HTML 태그 내에 이벤트 리스너 작성
 - ▣ HTML 태그의 이벤트 리스너 속성에 리스너 코드 직접 작성

예) <p>태그에 마우스 올리면 skyblue, 내리면 white로 배경변경

1. 이벤트

6

```
<h3>HTML 태그에 리스너 작성</h3>
<hr>
<p onmouseover="this.style.backgroundColor='skyblue'"
  onmouseout="this.style.backgroundColor='white'">
  마우스 올리면 하늘 색으로 변경</p>
```

HTML 태그에 리스너 작성

마우스 올리면 하늘 색으로 변경

1. 이벤트

7

□ DOM 객체의 이벤트 리스너 프로퍼티에 이벤트 리스너 코드 작성

예)

```
<p id= "p" >마우스 올리면 하늘색으로 변경</p>
```

```
function over() { // onmouseover 리스너로 사용할 함수  
    ...  
}
```

```
var p = document.getElementById("p");  
p.onmouseover = over;    // onmouseover 리스너로 over()  
함수 등록
```

```
p.onmouseover = over{}; // 잘못된 코드
```

1. 이벤트

8

```
<script>
var p;
function init() { // 문서가 완전히 로드되었을 때 호출
    p = document.getElementById("p");
    p.onmouseover = over; // over() 함수를 onmouseover 리스너로 등록
    p.onmouseout = out; // out() 함수를 onmouseout 리스너로 등록
}
function over() {
    p.style.backgroundColor="skyblue";
}
function out() {
    p.style.backgroundColor="white";
}
</script></head>
<body onload="init()">
<h3>DOM 객체의 이벤트 리스너 프로퍼티에 함수 등록</h3>
<hr>
<p id="p">마우스 올리면 하늘 색으로 변경</p>
</body>
</html>
```


1. 이벤트

9

□ DOM 객체의 `addEventListener()` 메소드 활용

```
addEventListener(eventName, listener[, useCapture])
```

- `eventName` : 이벤트 타입을 나타내는 문자열. `click`, `load`, `keydown` 등
- `listener` : 이벤트 리스너로 등록할 함수 이름
- `useCapture` : `true`이면 이벤트 흐름 중 캡처 단계에서 실행될 리스너(`listener` 함수) 등록.
`false`이면 버블 단계에서 실행될 리스너 등록. 생략 가능하며 디폴트는 `false`.
이벤트 흐름은 3절에서 자세히 설명

`listener` 함수를 `eventName`의 이벤트를 처리할 리스너로 등록한다.

```
p.addEventListener("mouseover", over); // onmouseover 리스너로 over() 등록
```

1. 이벤트

10

```
<script>
var p;
function init() { // 문서가 완전히 로드되었을 때 호출
    p = document.getElementById("p");
    p.addEventListener("mouseover", over); // 이벤트 리스너 등록
    p.addEventListener("mouseout", out); // 이벤트 리스너 등록
}
function over() {
    p.style.backgroundColor="skyblue";
}
function out() {
    p.style.backgroundColor="white";
}
</script>
</head>
<body onload="init()">
<h3>addEventListener()를 이용한 리스너 등록</h3>
<hr>
<p id="p">마우스 올리면 하늘 색으로 변경</p>
</body>
```

1. 이벤트

11

- 익명 함수로 이벤트 리스너 작성
 - ▣ 익명 함수(anonymous function)
 - 함수 이름 없이 필요한 곳에 함수의 코드를 바로 작성
 - ▣ 코드가 짧거나 한 곳에서만 사용하는 경우, 익명 함수 편리

```
p = document.getElementById("p");

p.onmouseover = function () { // 익명 함수
    this.style.backgroundColor = "skyblue";
};

p.addEventListener("mouseout",
    function () { this.style.backgroundColor="white"; } // 익명 함수
);
```

1. 이벤트

12

```
<script>
var p;
function init() { // 문서가 완전히 로드되었을 때 호출
    p = document.getElementById("p");
    p.onmouseover = function () { // 익명 함수
        this.style.backgroundColor = "skyblue";
    };
    p.addEventListener("mouseout",
        function () { this.style.backgroundColor="white"; } // 익명 함수
    );
}
</script>
</head>
<body onload="init()">
<h3>익명 함수로 이벤트 리스너 작성</h3>
<hr>
<p id="p">마우스 올리면 하늘 색으로 변경</p>
</body>
```

1. 이벤트

13

```
function over() {  
    p.style.backgroundColor="orchid";  
}
```

[1] HTML 태그

```
<p id="p"  
onmouseover="this.style.backgroundColor='orchid'"  
  마우스 올리면 orchid 색으로 변경  
</p>
```

[2] 이벤트 리스너 프로퍼티

```
function over() {  
    p.style.backgroundColor="orchid";  
}  
p.onmouseover = over;
```

[3] addEventListener() 메소드 이용

```
p.addEventListener("mouseover", over);
```

[4] 익명 함수 이용

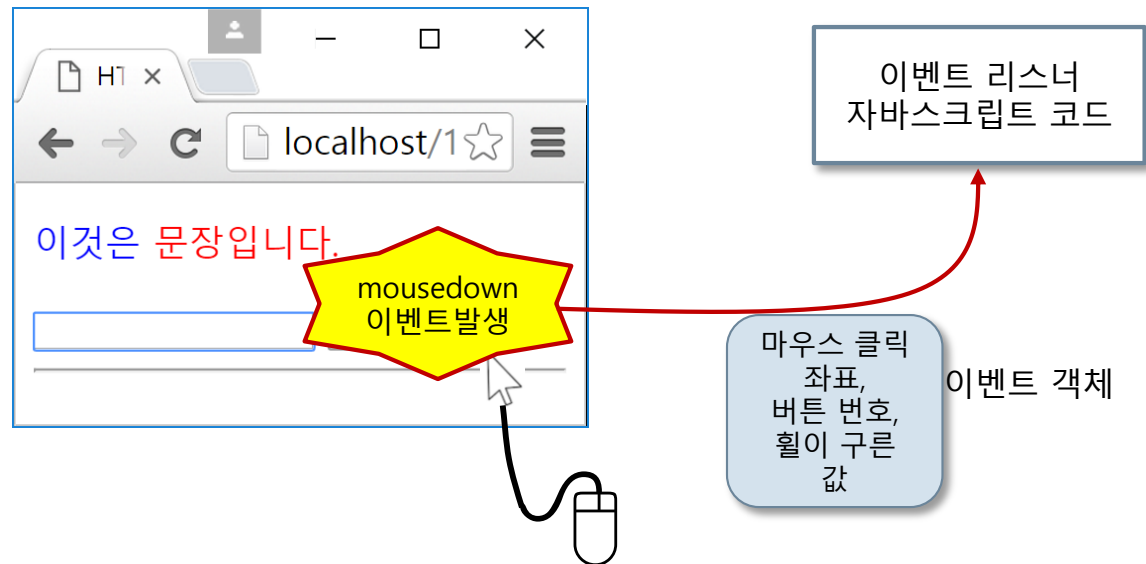
```
p.onmouseover = function ()  
{ this.style.backgroundColor="orchid"; };  
p.addEventListener("mouseover",  
  function () { this.style.backgroundColor="orchid"; }  
);
```

2. 이벤트 객체

14

□ 이벤트 객체(event object)

- 발생한 이벤트에 관련된 다양한 정보를 담은 객체
- 예) mousedown 이벤트의 경우, 마우스 좌표와 버튼 번호 등, keydown 이벤트의 경우, 키 코드 값 등
- 이벤트가 처리되고 나면 이벤트 객체 소멸



2. 이벤트 객체

15

- 이벤트 객체는 이벤트 리스너 함수의 첫 번째 매개변수에 전달

1. 이름을 가진 이벤트 리스너 함수

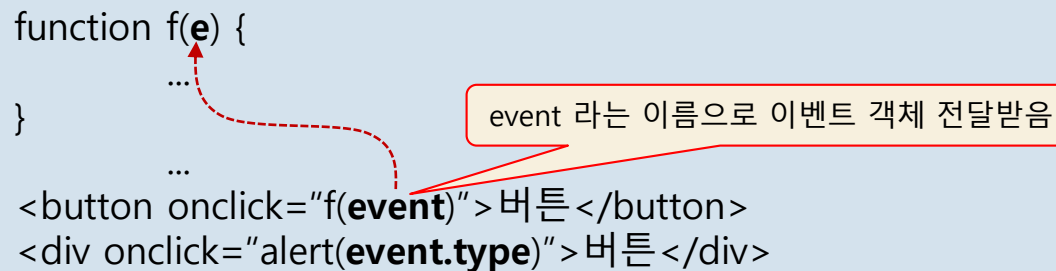
```
function f(e) { // 매개변수 e에 이벤트 객체 전달받음
    ...
}
obj.onclick = f; // obj 객체의 onclick 리스너로 함수 f 등록
```

2. 익명 함수의 경우

```
obj.onclick = function(e) { // 매개변수 e에 이벤트 객체 전달받음
    ...
}
```

3. HTML 태그에 이벤트 리스너 : event 라는 이름으로 전달

```
function f(e) {
    ...
}
...
<button onclick="f(event)">버튼</button>
<div onclick="alert(event.type)">버튼</div>
```



A red dashed arrow originates from the `event` text in the HTML attributes `onclick="f(event)"` and `onclick="alert(event.type)"`, pointing to the parameter `e` in the function definition `function f(e) {`. A red-bordered callout box with a yellow background contains the text "event 라는 이름으로 이벤트 객체 전달받음" (Received event object by the name event), with a red arrow pointing from the box to the `event` text in the first HTML attribute.

2. 이벤트 객체

16

```
<body>
<p id="p">마우스를 올려 보세요</p>
<button onclick="f(event)">클릭하세요</button>
<script>
function f(e) { // e는 현재 발생한 이벤트 객체
    alert(e.type); // 이벤트 종류 출력
}

document.getElementById("p").onmouseover = f;
</script>
</body>
```

마우스를 올려 보세요

클릭하세요

이 페이지 내용:

mouseover

확인

마우스를 올려 보세요

클릭하세요

이 페이지 내용:

click

확인

2. 이벤트 객체

17

- 이벤트 객체에 들어 있는 정보
 - ▣ 현재 발생한 이벤트에 관한 다양한 정보
 - 이벤트 객체의 프로퍼티와 메소드로 알 수 있음
 - ▣ 이벤트의 종류마다 조금씩 다름
 - 이벤트 객체의 공통 멤버

멤버	종류	설명
type	프로퍼티	현재 발생한 이벤트의 종류를 나타내는 문자열(click, load 등)
target	프로퍼티	이벤트를 발생시킨 객체(DOM 객체 혹은 HTML 태그)
currentTarget	프로퍼티	현재 이벤트 리스너를 실행하고 있는 DOM 객체
defaultPrevented	프로퍼티	이벤트의 디폴트 행동이 취소되었는지를 나타내는 true/false 값
preventDefault()	메소드	이벤트의 디폴트 행동을 취소시키는 메소드

- ▣ target 프로퍼티
 - 이벤트 타겟 객체 가리킴
 - 이벤트 타겟 : 이벤트를 유발시킨 DOM 객체
 - <button> 태그의 버튼을 클릭하였으면, 이때 click 이벤트의 이벤트 타겟은 버튼

2. 이벤트 객체

<h3>이벤트 객체의 프로퍼티 출력</h3>

<hr>

<p id="p">버튼을 클릭하면 이벤트 객체를 출력합니다.</p>

<button onclick="f(event)">클릭하세요</button>

<script>

function f(e) { // e는 현재 발생한 이벤트 객체

var text = "type: " + e.type + "
"

+ "target: " + e.target + "
"

+ "currentTarget: " + e.currentTarget + "
"

+ "defaultPrevented: " + e.defaultPrevented;

var p = document.getElementById("p");

p.innerHTML = text; // 이벤트 객체의 프로퍼티 출력

}

</script>

이벤트 객체의 프로퍼티 출력

버튼을 클릭하면 이벤트 객체를 출력합니다.

클릭하세요

이벤트 객체의 프로퍼티 출력

type: click

target: [object HTMLButtonElement]

currentTarget: [object HTMLButtonElement]

defaultPrevented: false

클릭하세요

2. 이벤트 객체

19

□ 이벤트의 디폴트 행동이란?

- 특정 이벤트에 대한 HTML 태그의 기본 행동
- 사례
 - <a>의 click 이벤트의 디폴트 행동 : 웹 페이지 이동
 - Submit 버튼의 click 이벤트의 디폴트 행동 : 폼 데이터 전송
 - <input type= "checkbox" >의 click 이벤트의 디폴트 행동 : 체크박스선택

□ 이벤트의 디폴트 행동을 막는 방법

□ 1. 이벤트 리스너에서 false 리턴

```
<a href="http://www.naver.com" onclick="return false">  
    이동 안되는 링크  
</a>
```

□ 2. 이벤트 객체의 preventDefault() 메소드 호출

```
<a href="http://www.naver.com" onclick="event.preventDefault();">  
    이동 안되는 링크  
</a>
```

□ 이벤트 객체의 cancelable 프로퍼티가 true인 경우만 취소 가능

```

<script>
function query() {
    var ret = confirm("네이버로 이동하시겠습니까?");
    return ret; // confirm()의 리턴 값은 true 또는 false
}

function noAction(e) {
    e.preventDefault(); // 이벤트의 디폴트 행동 강제취소
}
</script>
</head>
<body>
<h3>이벤트의 디폴트 행동 취소</h3>
<hr>
<a href="http://www.naver.com"
    onclick="return query()">
    네이버로 이동할 지 물어보는 링크</a>
<hr>
<form>
    <input type="checkbox">빵(체크 됨)<br>
    <input type="checkbox"
        onclick="noAction(event)">술(체크 안됨)
</form>
</body>

```

이벤트의 디폴트 행동 취소

네이버로 이동할 지 물어보는 링크

- ☐ 빵(체크 됨)
- ☐ 술(체크 안됨)

이 페이지 내용:

네이버로 이동하시겠습니까?

확인

취소

3. 이벤트 흐름

21

□ 이벤트 흐름이란?

- 이벤트가 발생하면 window 객체에 먼저 도달하고, DOM 트리를 따라 이벤트 타겟에 도착하고, 다시 반대 방향으로 흘러 window 객체에 도달한 다음 사라지는 과정

□ 이벤트가 흘러가는 과정

□ 캡처 단계(capturing phase)

- 이벤트가 window 객체에서 중간의 모든 DOM 객체를 거쳐 타겟 객체에 전달되는 과정
- 이벤트가 거쳐가는 모든 DOM 객체(window포함)의 이벤트 리스너 실행

□ 버블 단계(bubbling phase)

- 이벤트가 타겟에서 중간의 모든 DOM 객체를 거쳐 window 객체에 전달되는 과정
- 이벤트가 거쳐가는 모든 DOM 객체(window포함)의 이벤트 리스너 실행

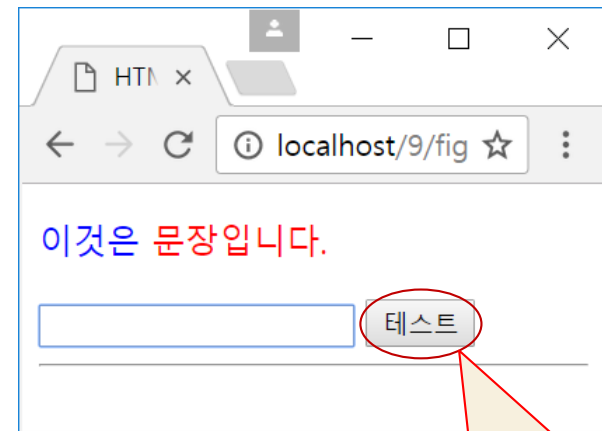
□ DOM 객체에는 캡처 리스너와 버블 리스너 두 개 모두 작성할 수 있음

3. 이벤트 흐름

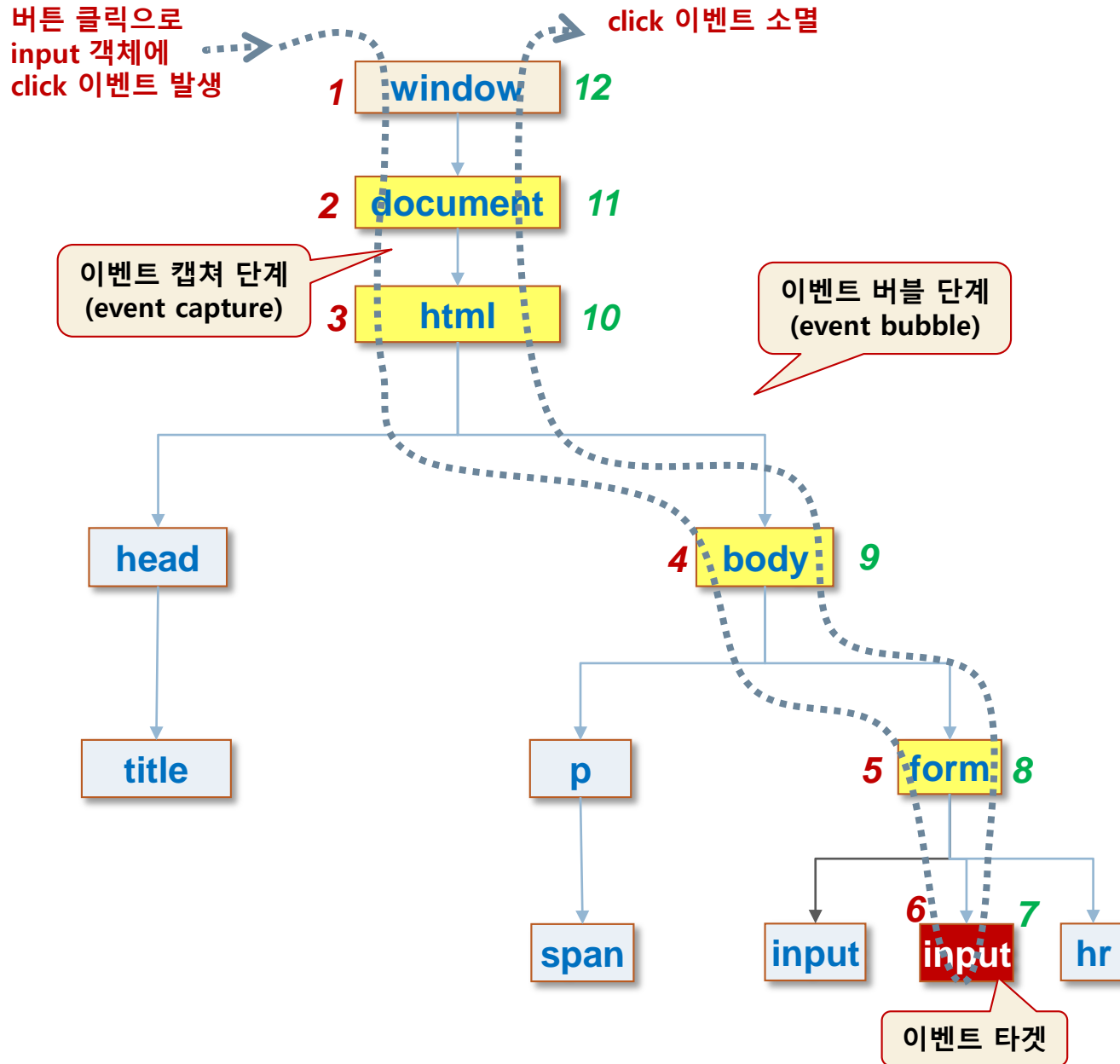
22

□ 샘플 웹 페이지

```
<!DOCTYPE html>
<html> <head> <title>HTML DOM 트리
</title> </head>
<body>
<p style="color:blue" >이것은
  <span style="color:red">문장입니다.</span>
</p>
<form>
  <input type="text">
  <input type="button" value="테스트"
id="button">
  <hr>
</form>
</body> </html>
```



버튼 클릭, click 이벤트 발생



3. 이벤트 흐름

24

- DOM 객체의 이벤트 리스너
 - 캡처 리스너와 버블 리스너를 모두 소유 가능
 - 이벤트 리스너 등록 시, 캡처 리스너인지 버블 리스너인지 구분
- 캡처 리스너와 버블 리스너 등록
 - `addEventListener()`의 3 번째 매개 변수 이용
 - `true`이면 캡처 리스너, `false`이면 버블 리스너

```
var b = document.getElementById("button");  
b.addEventListener("click", capFunc, true); // 캡처 단계에서 capFunc() 실행  
b.addEventListener("click", bubbleFunc, false); // 버블 단계에서 bubbleFunc() 실행
```

- 다른 방법의 이벤트 리스너 등록의 경우
 - 버블 리스너로 자동 등록
 - 예)

```
obj.onclick = function(e) { // 버블 리스너도 작동  
    ...  
}
```


3. 이벤트 흐름

25

이것은 문장입니다.

테스트

capture 단계 : BODY 태그 click이벤트
bubble 단계 : INPUT 태그 click이벤트
bubble 단계 : BODY 태그 click이벤트

```
<body>
<p style="color:blue">이것은
  <span style="color:red" id="span">문장입니다.
</span>
</p>
<form>
  <input type="text" name="s">
  <input type="button" value="테스트" id="button">
  <hr>
</form>
<div id="div" style="color:green"></div>
```

```
<script>
var div = document.getElementById("div"); // 이벤트 메시지 출력 공간
var button = document.getElementById("button");

2 // body 객체에 캡처 리스너 등록
document.body.addEventListener("click", capture, true); // 캡처 단계(1)

// 타겟 객체에 버블 리스너 등록
button.addEventListener("click", bubble, false); // 버블 단계(2)

// body 객체에 버블 리스너 등록
document.body.addEventListener("click", bubble, false); // 버블 단계(3)

function capture(e) { // e는 이벤트 객체
    var obj = e.currentTarget; // 현재 이벤트를 받은 DOM 객체
    var tagName = obj.tagName; // 태그 이름
    div.innerHTML += "<br>capture 단계 : " + tagName + " 태그 " + e.type + "
    이벤트";
}

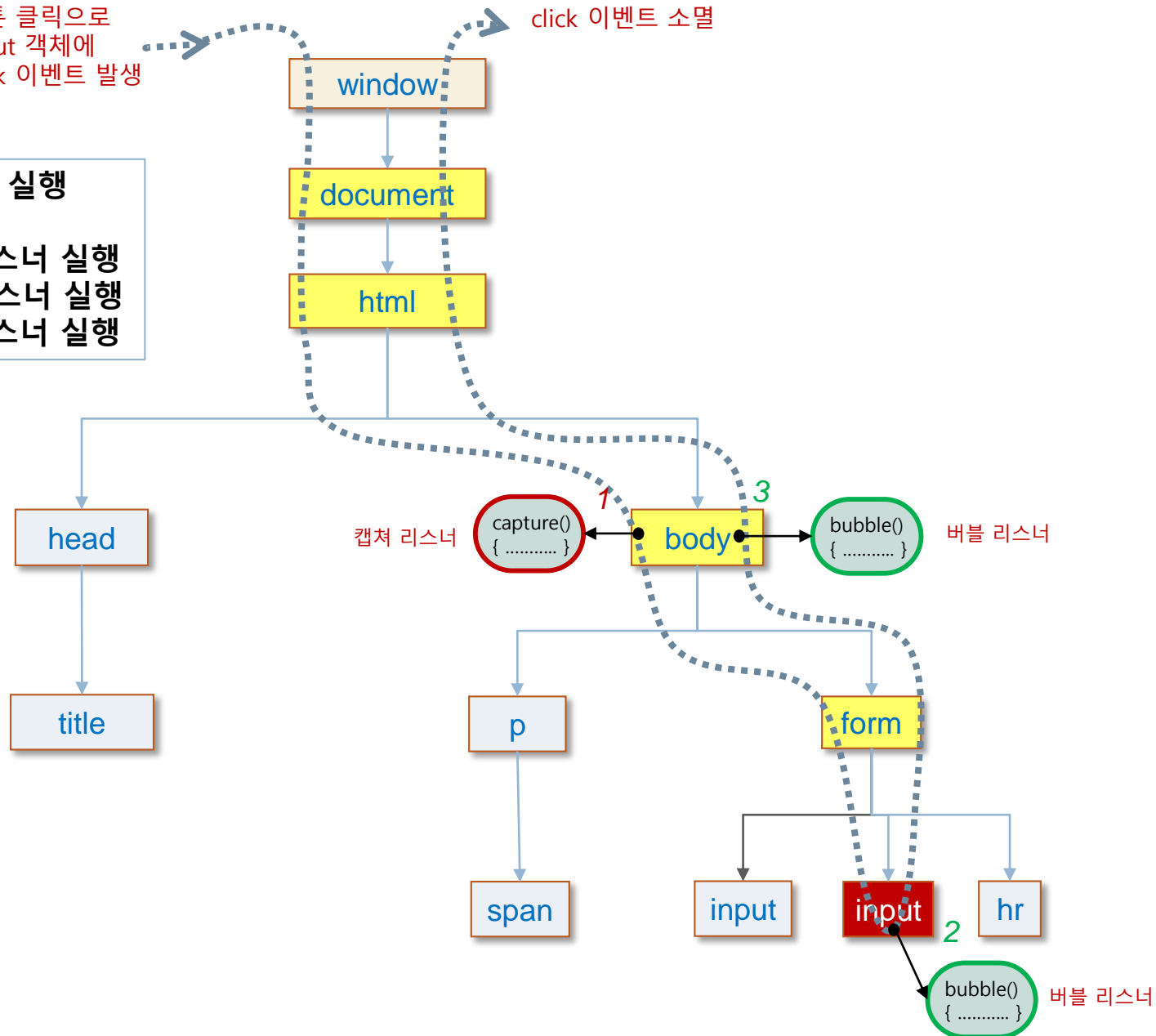
function bubble(e) { // e는 이벤트 객체
    var obj = e.currentTarget; // 현재 이벤트를 받은 DOM 객체
    var tagName = obj.tagName; // 태그 이름
    div.innerHTML += "<br>bubble 단계 : " + tagName + " 태그 " + e.type + "
    이벤트";
}
</script>
```

버튼 클릭으로
input 객체에
click 이벤트 발생

click 이벤트 소멸

웹 페이지의 이벤트 리스너 실행

1. <body> 태그의 캡처 리스너 실행
2. <input> 태그의 버블 리스너 실행
3. <body> 태그의 버블 리스너 실행



3. 이벤트 흐름

28

- 이벤트 흐름을 중단시킬 수 있는가?
 - YES
- 이벤트 객체의 stopPropagation() 호출
 - `event.stopPropagation();` // event가 이벤트 객체일 때