9장

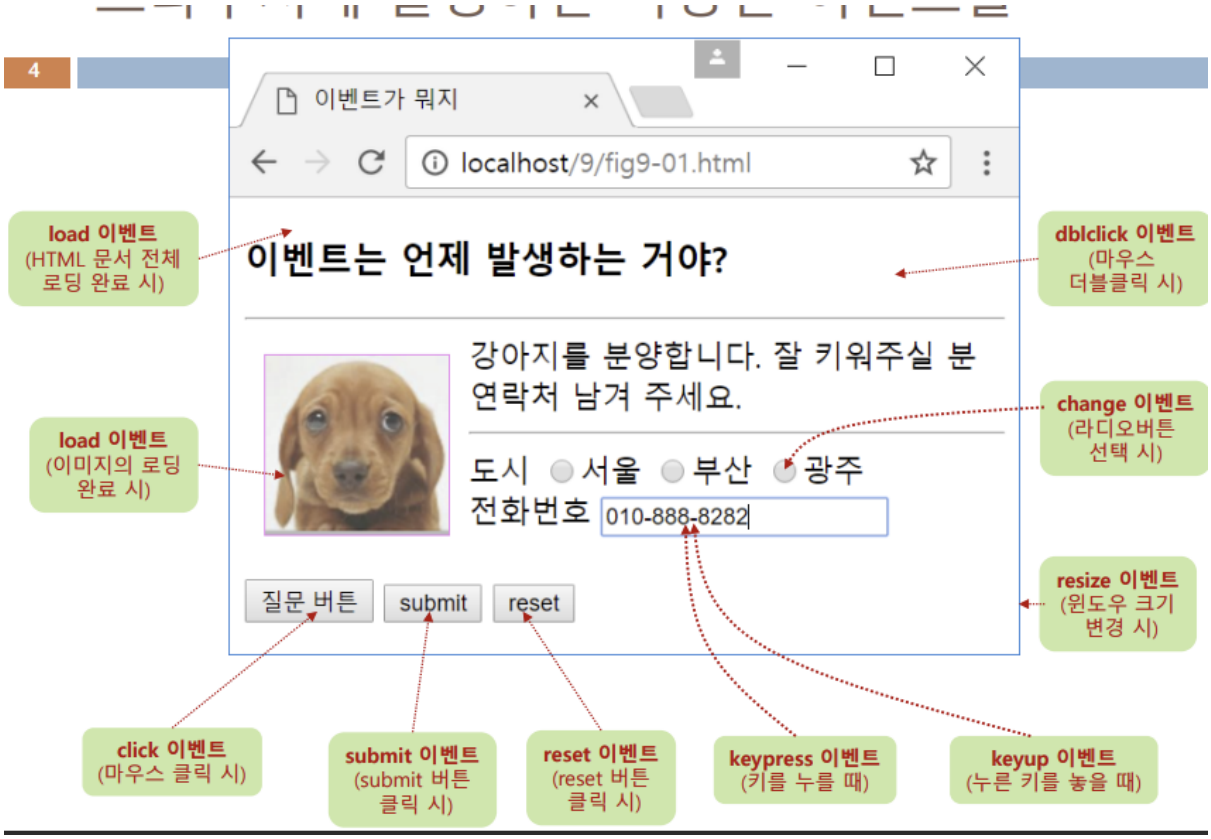
이벤트 기초 및 활용

- 이벤트 개요

→ 이벤트 : 마우스 클릭, 키보드 입력, 이미지나 html 문서의 로딩, 타이머의 타임아웃 등 사용자의 입력 행위나 문서나 브라우저의 상태 변화를 자바스크립트 코드에게 알리는 통지

→ 이벤트 리스너 : 발생한 이벤트에 대처하기 위해 작성된 자바스크립트 코드

→ 이벤트 종류 : 70여 가지



- 이벤트 리스너 만들기

1. html 태그 내에 작성
2. dom 객체의 이벤트 리스너 프로퍼티에 작성
3. dom 객체의 addeventlistener() 메소드 이용

- html 태그 내에 이벤트 리스너 작성

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

- DOM 객체의 이벤트 리스너 프로퍼티에 작성

```
<p id="p">마우스 올리면 orchid 색으로 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
```

- DOM 객체에 addEventListner() 메소드 활용

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

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

listener 함수를 eventName의 이벤트를 처리할 리스너로 등록한다.

예)

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

- 익명 함수로 이벤트 리스너 작성

→ 익명 함수 : 함수 이름 없이 필요한 곳에 함수의 코드를 바로 작

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

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

==⇒ 코드가 짧거나 한 곳에서만 사용하는 경우에 편리하다.

<4가지 방법 비교>

```
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"; };
```

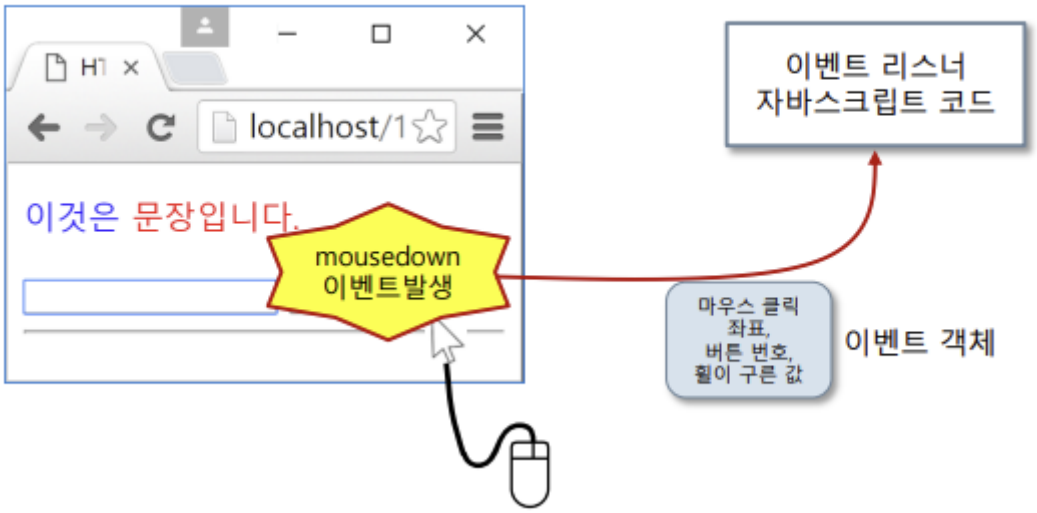
(5) 익명 함수 이용

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

- 이벤트 객체

: 발생한 이벤트에 관련된 다양한 정보를 담은 객체

: 이벤트가 처리되고 나면 이벤트 객체 소멸



- 이벤트 객체 전달받기

1. 이름을 가진 이벤트 리스너

```
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 : event라는 이름으로 전달

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

event 라는 이름으로 이벤트 객체 전달받음

- 이벤트 객체에 들어있는 정보

: 현재 발생한 이벤트에 대한 다양한 정보

: 이벤트의 종류마다 조금씩 다름

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 (ex : <button> 태그의 버튼을 클릭하였으면, 이때 click 이벤트의 이벤트 타켓은 버튼이다.)

- 이벤트의 디폴트 행동 취소 : preventDefault()

⇒ 이벤트의 디폴트 행동이란? → 특정 이벤트에 대한 html 태그의 기본 행동

⇒ 이벤트의 디폴트 행동을 막는 방법

1) 이벤트 리스너에서 false 리턴

2) 이벤트 객체의 preventDefault() 메소드 호출

⇒ 이벤트 객체의 cancelable 프로퍼티가 true인 경우만 취소 가능

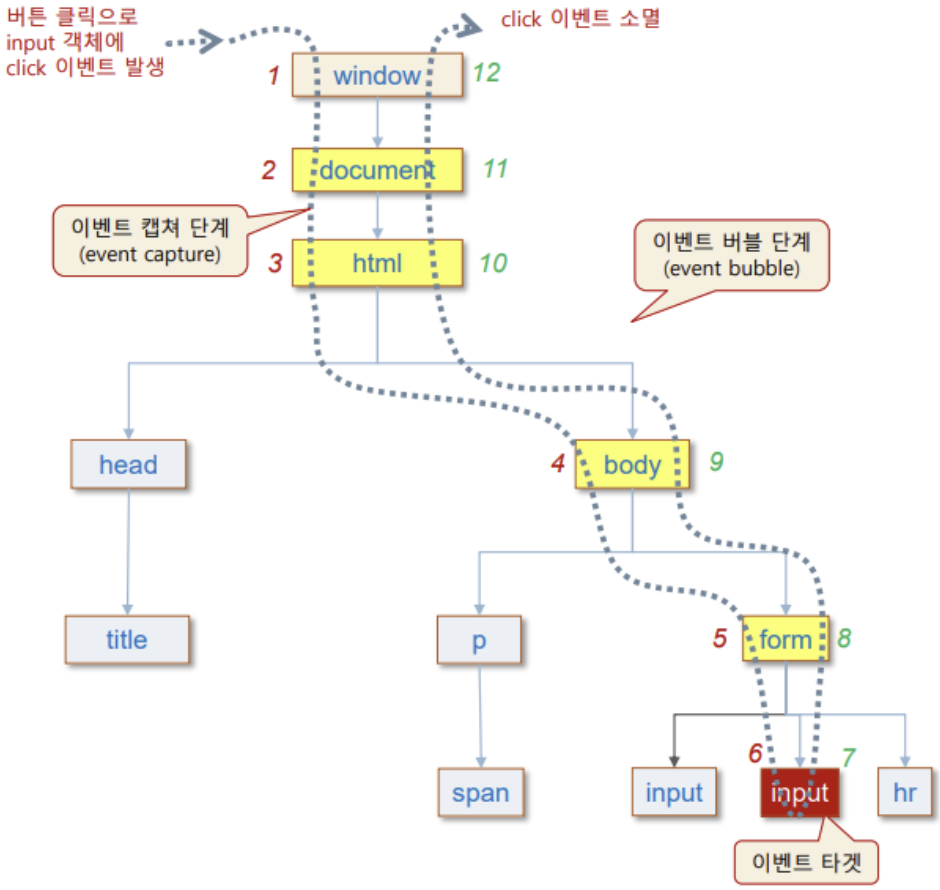
- 이벤트 흐름

: 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

캡쳐 단계(window → DOM) → 버블 단계(DOM → window)

ex)



- 캡쳐 리스너와 버블 리스너

⇒ DOM 객체의 이벤트 리스너 : 캡쳐 리스너와 버블 리스너를 모두 소유 가능

⇒ 캡쳐 리스너와 버블 리스너의 등록 ⇒ addEventListener()의 3번째 매개변수 이용

: true : 캡쳐 리스너 / false : 버블 리스너

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

: 다른 방법으로 이벤트 리스너를 등록 하면 자동으로 버블 리스너!

- 이벤트 흐름의 중단 : 이벤트 객체의 stopPropagation() 호출

- 마우스 핸들링

⇒ 마우스 이벤트 객체의 프로퍼티

프로퍼티	
x, y	(x, y)는 타겟 객체의 부모 객체 내에서의 마우스 좌표
clientX, clientY	(clientX, clientY)는 브라우저 윈도우의 문서출력 영역 내에서의 마우스의 좌표
screenX, screenY	(screenX, screenY)는 스크린을 기준으로 한 마우스 좌표
offsetX, offsetY	(offsetX, offsetY)는 타겟 객체 내에서의 마우스 좌표
button	눌러진 마우스 버튼 • 0 : 아무 버튼도 눌러지지 않았음 • 1 : 왼쪽 버튼이 눌러졌음 • 2 : 오른쪽 버튼이 눌러졌음 • 3 : 왼쪽, 오른쪽 버튼이 모두 눌러졌음 • 4 : 중간 버튼이 눌러졌음
wheelDelta	마우스 휠이 구른 방향 • 양수 : 위쪽으로 굴린 경우(실제 wheelDelta 값은 120) • 음수 : 아래쪽으로 굴린 경우(실제 wheelDelta 값은 -120)

*onclick : HTML 태그가 클릭 될 때 / ondbclick : HTML 태그가 더블 클릭 될 때

*ondblclick : HTML 태그가 더블 클릭 될 때

- 여러 마우스 관련 이벤트 리스너

- onmousedown : 마우스 버튼을 누르는 순간
- onmouseup : 눌러진 버튼이 놓여지는 순간
- onmouseover : 마우스가 태그 위로 올라오는 순간. 자식 영역 포함
- onmouseout : 마우스가 태그 위로 올라오는 순간. 자식 영역 포함
- onmouseenter : 마우스가 태그 위로 올라오는 순간. 버블 단계 없음
- onmouseleave : 마우스가 태그 위로 올라오는 순간. 버블 단계 없음
- onwheel : HTML 태그에 마우스 휠이 구르는 동안 계속 호출

→ 위쪽으로 굴린 경우 : wheelDelta > 0 / 아래쪽으로 굴린 경우 : wheelDelta < 0

-onmousemove: 마우스의 위치를 보여

→ clientX :마우스 포인터의 뷰 포트(Viewport) 상의 X 좌표

→ clientY : 마우스 포인터의 뷰 포트(Viewport) 상의 Y 좌표

→ screenX : 마우스 이벤트가 발생한 X 좌표를 스크린(Screen)의 기준으로 나타냅니다.

→ screenY : 마우스 이벤트가 발생한 Y 좌표를 스크린(Screen)의 기준으로 나타냅니다.

→ offsetX 마우스 이벤트가 발생한 X 좌표를 이벤트 대상 요소(Element)의 상대적인 위치로 나타냅니다.

→ offsetY 마우스 이벤트가 발생한 Y 좌표를 이벤트 대상 요소(Element)의 상대적인 위치로 나타냅니다.

- oncontextmenu

: HTML 태그 위에 마우스 오른쪽 버튼 클릭 → 디폴트로 컨텍스트 메뉴(context menu) 출력

⇒ '소스 보기' 나 '이미지 다운로드' 등의 메뉴

: oncontextmenu 리스너가 먼저 호출 → falese를 리턴하면 컨텍스트 메뉴를 출력하는 디폴트 행동 취소

```
document.oncontextmenu = function () {
  ...
  return false; // 컨텍스트 메뉴 출력 금지
}
```

- 문서의 로딩 완료와 onload

: onload → window 객체에 발생 : 웹 페이지의 로딩 완료 시 호출되는 이벤트 listener

: onload listener 작성 방법

1. window.onload='alert('onload');"
2. <body onload="alert('onload');">

⇒ 이 둘은 같은 표현이다.

- 이미지 로딩 완료와 onload

:image객체 → <img> 태그에 의해 생성되는 DOM 객체, new Image() → 자바스크립트 코드에 의해 생성되는 객체

:onload → 이미지의 로딩이 완료되면 Image 객체에 발생하는 이벤트

: 새로운 이미지를 로딩하는 방법

```
<img id="myImg" src="apple.png" width="..." height="...">

var myImg = document.getElementById("myImg");
myImg.src = "banana.png";
```

banana.png 이미지의 로딩이 완료된 myImg의 onload 리스너 실행

**주의할 점 !

로딩 완료 시점을 잘 고려해야 한다.

- new Image()로 이미지 로딩과 출력

⇒ new Image()

```
var bananaImg = new Image();      // 이미지 객체 생성
bananaImg.src = "banana.png";     // 이미지 로딩
```

⇒ 로딩 된 이미지 출력

```
<img id="myImg" src="apple.png" width="..." height="...">

var myImg = document.getElementById("myImg");
myImg.src = bananaImg.src;        // 이미지 출력
```

- onblur와 onfocus

: 포커스 - 포커스는 현재 키 입력에 대한 독점권, 브라우저는 포커스를 가지고 있는 HTML 태그 요소에 키 공급

: onblur - 포커스를 을 때 발생하는 이벤트 리스너

: onfocus - 포커스를 잃을 때 발생하는 이벤트 리스너

- 라디오 버튼과 체크 박스

: 라디오 버튼 객체

: 체크 박스 객체

// 이미 아는 거^^

- select 객체와 onchange

: select 객체 : <select> 태그로 만들어진 콤보 박스

→ 선택된 옵션 알아내기

```
var sel = document.getElementById("fruits");
var index = sel.selectedIndex;     // index는 선택 상태의 옵션 인덱스
```

→ 옵션 선택

```
sel.selectedIndex = 2;              // 3번째 옵션 "사과" 선택
sel.options[2].selected = true;     // 3번째 옵션 "사과" 선택
```

→ 선택된 옵션이 변경되면 select 객체의 onchange 리스너 호출

*selectedIndex : 선택 상대의 옵션의 인덱

- 키 이벤트

: onkeydown → 키가 눌러지는 순간 호출 , 모든 키에 대해 작동

: onkeypress → 문자 키와 <enter> , <space> , <esc> 키에 대해서만 눌러지는 순간에 추가 호출

: onkeyup → 눌러진 키가 떼어지는 순간 호출

- onreset과 onsubmit

: onreset → reset 버튼(<input type="reset">) 클릭 시 false를 리턴 하면 폼이 초기화되지 않음

: onsubmit → submit 버튼 클릭 시 false를 리턴 하면 폼 전송하지 않