1 이벤트의 이해

- **이벤트** : 웹 브라우저에서 웹 문서에 특별한 일이 있을 때 발생하는 신호
- **이벤트 핸들러** : DOM 객체에 할당되어 해당 객체의 이벤트 반응에 호출되어 처리되는 프로퍼티
- **이벤트 모델의 종류**
  - 기본 이벤트 모델(original event model) – DOM Level 0 이벤트 모델
  - 표준 이벤트 모델(standard event model) – DOM Level 2 이벤트 모델
  - 인터넷익스플로러 이벤트 모델(Internet Explorer event model)

기본 이벤트 모델

- **기본 이벤트 모델의 이벤트 핸들러**
  - 마우스 관련 : onclick, onmousedown, onmouseup, onmouseover, onmouseout
  - 관련 : onload, onunload
  - 서식 관련 : onsubmit, onreset, onfocus, onblur
  - 키보드 관련 : onkeydown, onkeypress, onkeyup
- **기본 이벤트 모델 적용**
  - HTML 요소에 속성으로 이벤트 핸들러 적용
    - 〈a href=" #" onclick=" var i=3, j=5; calc=i*j; alert(calc);" 〉3 x 5 의 값은?〈/a〉
    - 〈body onload=" popup();" 〉
  - 자바스크립트 DOM 객체 프로퍼티로 이벤트 핸들러와 호출함수 지정
    - window.onload = init;
    - document.getElementById("next").onclick = goNext;
  - 이벤트 기본 기능 막기
    - 웹 브라우저에 기본으로 정의 되어 있는 이벤트 기능의 작동을 막을 필요성 있음
    - 기본 이벤트 모델에서는 이벤트 핸들러 호출 함수가 false를 리턴하면 기본 기능이 작동을 하지 않음
    - 〈a href=" second.html" onclick=" alert('두 번째 페이지로 갈까요?'); return false;" 〉Second Page〈/a〉
      위 코드는 'second.html' 로 이동하지 않는다.
  - 이벤트 API
    - 이벤트 핸들러 핸들러는 호출함수에 함수 인자로 이벤트 객체를 전달한다.
    - 이벤트 객체를 전달받은 호출 함수는 다양한 이벤트 API를 이용하여 이벤트 정보를 얻을 수 있다.
    - 이벤트 발생 좌표 정보: clientX, clientY, screenX, screenY,
    - 이벤트 발생 키보드 정보: keyCode, altKey, shiftKey, type

2 표준 이벤트 모델

특징

- 표준화 된 기능 제공
- 고급 이벤트 처리 방식
- 고급 이벤트 처리 방식

이벤트 핸들러 적용

- window.addEventListener('load', init, false);.
  - 이벤트를 감지하고자 하는 객체에 프로퍼티로 addEventListener를 적용
  - 첫 번재 인자 'load' 는 이벤트 핸들러
    기본 이벤트 모델의 이벤트 핸들러에 앞에 on을 제거하고 사용
  - 두 번째 인자 'init' 는 호출 함수
  - 세 번째 인자 false는 이벤트 전파 방법

- 적용 예
  - document.getElementById("drag").addEventListener('mousedown', mouseDown, false);
  - document.getElementsByTagName("a")[0].addEventListener(click, linkClick, false);

이벤트 기본 기능 막기

- 표준 이벤트 모델에서는 preventDefault();이벤트 API 제공
- function mouseDown (event) {
  event.preventDefault();
  }

이벤트 API

- 기본 이벤트 모델에 비해 많은 수의 이벤트 API 제공
- 기본 이벤트 모델의 이벤트 API와 동일한 사용법
- target : 이벤트 발생 객체
  - event.target.style.display = "none" ;
- eventType : 발생한 이벤트타입
- timeSamp : 밀리초 단위의 이벤트 발생 시간(날짜와 시간)
- Button : 마우스 이벤트 발생시 눌려진 버튼.
  1은 왼쪽, 2은 가운데, 3은 오른쪽 버튼

this : 이벤트 핸들러 호출 함수에서 사용되며 이벤트 발생 객체를 의미함

- 이벤트 API인 target과 동일한 결과
- 함수를 호출하는 기본 이벤트 모델과 표준 이벤트 모델에서 모두 사용 가능
- 표준 이벤트 모델에서는 이벤트 전파로 인한 혼란으로 target을 사용하는 것을 권장함

적용된 이벤트 핸들러 함수의 제거

- removeEventListener : 적용된 이벤트 핸들러 함수를 제거하는데 사용한다.
- 이벤트 핸들러 호출 함수 적용 시 사용되었던 addEventListener와 동일한 인자로 구성된다.
  - document.getElementById("apple").addEventListener('click', bang, true);
  - document.getElementById("apple").removeEventListener('click', bang, true);

3 이벤트 전파

이벤트 전파의 이해

- 이벤트는 발생한 객체에만 머물지 않고 이벤트 발생 객체를 포함하는 모든 상위 객체에 전파된다.
- 이벤트 전파는 시각적으로 겹쳐 있는 객체에서 일어나는 것이 아니라 부모 자식 관계로 연결된 노드에서 발생하는 것.

이벤트 전파의 3단계

- 1단계 : document 노드 에서 이벤트 발생 노드까지 내려가면서 이벤트 전파
- 2단계 : 이벤트 발생 노드
- 3단계 : 이벤트 발생 노드에서 document 노드까지 올라가면서 이벤트 전파

addEventListener 이벤트 전파 조절

- addEventListener의 세번째 인자를 참으로 해 놓으면 이벤트 전파의 전체 과정에서 이벤트 핸들러가 이벤트를 감지한다.–캡춰링capturing
- 거짓이면 2단계 3단계에서 이벤트 핸들러가 이벤트를 감지한다. – 버블링bubbling

이벤트 전파 방지

- stopPropagation : 이벤트 전파를 막는 이벤트 API

이벤트 전파에 관련된 이벤트 API

- currentTarget : 이벤트 전파로 인해 현재 도달한 객체
- eventPhase: 세 단계의 이벤트 전파 과정 중 현 단계가 몇 단계인지 알려준다.