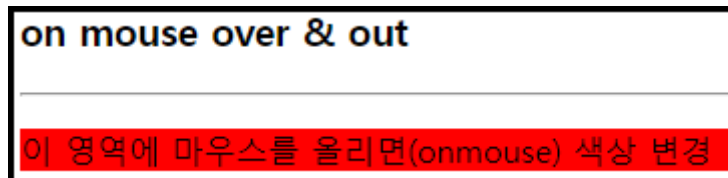


1] onmouseover & onmouseout : 마우스 액션 스크립트

- onmouseover : 마우스의 커서가 특정 영역에 도달시 적용할 기능
- onmouseout : 마우스의 커서가 특정 영역에서 벗어날시 적용할 기능
 - HTML의 <head><script> 영역 혹은 <body> 영역에서만 작성

☐ 구현 목표 : 아래의 이미지



마우스를 올리면 *red*, 벗어나면 *white*를 적용

- ☐ 오늘의 코드는 자바스크립트로 옮기는 것이 좋으나, HTML에서만 코드를 작성하면 한 화면에서 기능을 이해하기 편리하기 때문
- ☐ 가급적 function a,b,c 로 함수명을 통일한다. 이는 내장함수와 구분하기 위함
- ☐ 금일의 학습과정에서는 5개 정도의 코드로 위 이미지와 동일한 결과를 구현
- ☐ 5개의 기능을 학습하는데 중점, 5개중 하나를 구현가능해도 무방

1) onmouseover & onmouseout 코드의 직접 작성

```
<body>
  <h3>on mouse over & out</h3><hr>
  <p onmouseover="this.style.backgroundColor='red'"
    onmouseout="this.style.backgroundColor='white'"
  >이 영역에 마우스를 올리면(onmouse) 색상 변경</p>
</body>
```

- 5개 중 유일하게 <body> 에서 작성을 종료
- tag의 내용이 길어지는 단점이 있다...고는 하나 단순 길이로는 가장 짧은 편
- BOM, DOM이 문제?, 이유는 설명이 부족

- ❖ 다른 4가지 방법은 공통적으로 <body> 에 아래의 코드를 작성
- ❖ <head> 의 <script> 영역에서의 코드만 다소 차이가 있는 편

```
<body onload="init()">
  <h3>리스너 작성</h3><hr>
  <p id="p"> 마우스를 올리면(onmouse) 색상 변경</p>
</body>
```

2) 각각의 function 을 지정

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <script>
      function a(){
        p=document.getElementById("p");
        p.onmouseover = b;
        p.onmouseout = c;
        function b(){p.style.backgroundColor="red";}
        function c(){p.style.backgroundColor="white";}
      }
    </script>
  </head>

  <body onload="a()">
    <h3>리스너 작성</h3><hr>
    <p id="p"> 이 영역에 마우스를 올리면(onmouse) 색상 변경</p>
  </body>
</html>
```

전체 화면, 이후의 설명에서는 `<body>` 부분이 중복이므로 생략하고, `head`의 `script` 부분만 작성하겠음

- `let p` : 변수의 선언, 없어도 자바스크립트는 허용, 본문에 `p`가 많이 사용되므로 편의상 선언함
- `function init() : init()`이라는 내장함수는 없음¹, 편의상 `a`로 대체
- `document.getElementById("p")`
 - `document` : 현재 (HTML) 문서 '의'
 - `getElementById("p")` : `id="p"`가 지정된 곳의 내용을 가져오겠다.
 - `getElement` : 해당되는 요소를 가져오겠다
 - `ById` : `id`로 지정된 곳에서
 - `("p")` : 본문에서는 `id="p"`를 의미, `p`가 위치한 곳에 기능을 적용하기 위함
- `p.onmouseover = b` : `p`의 영역에 마우스가 올라오면 `b` 함수의 내용을 적용,
`p.onmouseout = c` : `p`의 영역에 마우스가 올라오면 `c` 함수의 내용을 적용
 - `function (b)` : 마우스를 올리면 `red` 색상 적용
 - `function (c)` : 벗어나면 `white` 색상 적용
- `<p id="p">` : 해당 태그로 인하여 `<p>` 함수의 영역과 함수 `b, c`의 색상 기능을 연결
 - `<p>` 함수의 영역 : 브라우저의 "마우스를 올리면(onmouse) 색상 변경"을 의미
- `onload`² : 문서의 모든 콘텐츠(images, script, css, etc)가 로드된 후 발생하는 이벤트
 1. `<head><script>` 부분을 임시로 읽고
 2. `<body>`의 내용을 포함한 모든 콘텐츠를 로드후
 3. `<onload>` 부분을 시행

¹ [객체 초기자](#) : `new object()`와 `object.create()`를 사용 - Mdn web docs

² [\[Javascript\] onload, ready](#)

3) addEventListener 를 활용한 지정

```
<script>
  let p;
  function a(){
    p=document.getElementById("p");
    p.addEventListener("mouseover", b); //p.onmouseover = b; 와 동일
    p.addEventListener("mouseout", c); //p.onmouseout = c; 와 동일
    function b(){p.style.backgroundColor="red";}
    function c(){p.style.backgroundColor="white";}
  }
</script>
```

- .addEventListener("mouseover", b)
- .addEventListener('이벤트타입', 실행할 함수)
- .addEventListener() : (괄호안의 내용)을 듣고(listener) 처리(add)하겠다
 - 이벤트가 발생했을때 그 처리를 담당하는 함수로 이벤트 핸들러라고도 함
 - 지정된 타입의 이벤트가 특정 요소에서 발생하면, 웹 브라우저는 그 요소에 등록된 이벤트 리스너를 실행
- mouseover : 이벤트타입, '마우스를 올리면' 을 의미
- b : 실행할 함수명
- .addEventListener는 소스코드가 많아질 경우 사용하는 것이 권장

4) 참조변수 this

```
<script>
  let p;
  function a(){
    p=document.getElementById("p");
    p.onmouseover=function(){this.style.backgroundColor="red"};
    p.onmouseout=function(){this.style.backgroundColor="white"};
  }
</script>
```

- 참조변수 this : 자기자신을 포함한 가장 가까운 객체를 호출(현재 a 를 의미)
- 앞서 function b,c 를 선언하지 않고 비교적 간략히 구성됨

5) EventListener + 참조변수 this

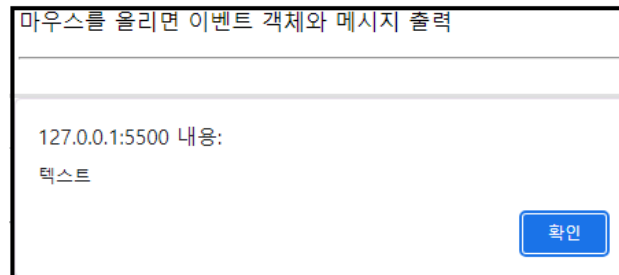
```
<script>
  let p;
  function a(){
    p=document.getElementById("p");
    p.addEventListener("mouseover", function(){this.style.backgroundColor="red"});
    p.addEventListener("mouseout", function(){this.style.backgroundColor="white"});
  }
</script>
```

- .addEventListener('이벤트타입', 실행할 함수)
 - 이벤트 타입 : mouseover 가 적용
 - 실행할 함수 : 참조변수 this를 활용

2] Alert, Confirm : 알림 스크립트

1) alert : event가 발생시 이벤트 객체(메시지를 담은 작은 이벤트 창)가 등장

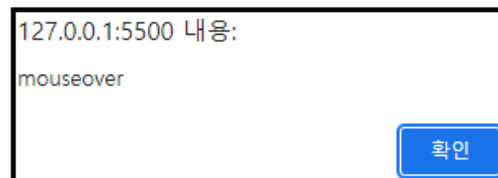
```
<body>
  <p id="p">마우스를 올리면 이벤트 객체와 메시지 출력</p><hr>
  <script>
    function a(){alert("텍스트");}
    document.getElementById("p").onmouseover= a;
  </script>
</body>
```



메시지("텍스트")가 담긴 이벤트 창이 열린다

- document.getElementById("p").onmouseover=a;
 - document.getElementById("p") : 본 문서의 p 영역에
 - .onmouseover = 마우스를 올리면
 - =a : a 함수를 실행, 이때 a 함수는 alert창과 텍스트가 출력
- function a(){alert("텍스트");}
 - 함수 a 를 호출시 alert 창이 뜨고 해당 창에 텍스트가 출력
- function a(b){alert("b.type");}

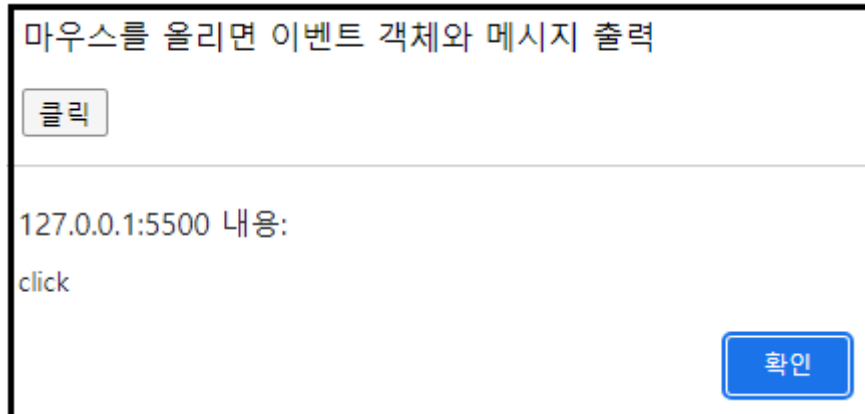
```
<body>
  <p id="p">마우스를 올리면 이벤트 객체와 메시지 출력</p>
  <script>
    function a(b){alert(b.type);}
    document.getElementById("p").onmouseover=a;
  </script>
</body>
```



- 이전에는 "텍스트"가 출력되었지만 현재는 mouseover 가 뜬
- function a(b){alert(b.type);}
 - 파라미터 (b)와 b.type가 연결,
 - type : 현재 발생하는 이벤트가 어떤 것인지 구분
 - mouseover에 의한 것임을 출력

- 버튼을 이용한 alert : 버튼(클릭)을 누르면 이벤트창과 type 로 인한 click 메시지 출력

```
<body>
  <p id="p">마우스를 올리면 이벤트 객체와 메시지 출력</p>
  <button onclick="a(event)">클릭</button>
  <script>
    function a(b){alert(b.type);}
    document.getElementById("p").onmouseover=a;
  </script>
</body>
```



- onclick="a(event)" : 사용자가 클릭시 a 함수의 기능이 담긴 이벤트가 발생
 - onclick : 브라우저에서 사용자가 클릭시
 - event : 이벤트가 발생, 일반적인 매개변수가 아닌 내장함수로서의 event³
 - a(event) : a 함수의 이벤트를 발생시켜라

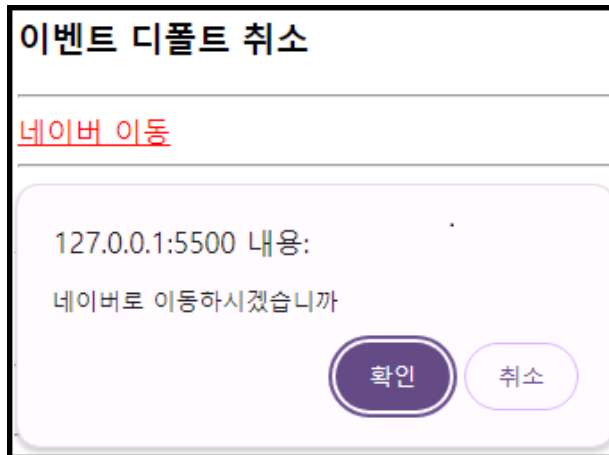
```
<body>
  <p id="p">마우스를 올리면 이벤트 객체와 메시지 출력</p>
  <button onclick="f(event)">클릭</button>
  <script>
    function f(e){alert(e.type);}
    document.getElementById("p").onmouseover=f;
  </script>
</body>
```

- function f(e) : a,b,c 로 해도 상관없으나 f(event)와 연결된 함수란 의미로 e 를 사용하는 것이 관례

³ [JavaScript - 이벤트\(Event\), 이벤트의 종류, 이벤트 연결](#)

- 2) confirm : event가 발생시 이벤트 객체(메시지를 담은 작은 이벤트 창)이 등장,
유저의 선택에 따라 확인(True) 와 취소(False) 의 연결값을 실행

```
<body>
  <h3>이벤트 디폴트 취소</h3><hr>
  <a href="https://www.naver.com" onclick="return a()">네이버 이동</a><hr>
</body>
<script>
  function a(){let b = confirm("네이버로 이동하시겠습니까"); return b;}
</script>
```



- confirm("텍스트") : 알림 스크립트, "텍스트"가 담긴 확인 및 취소 선택창 호출
 - 기본값으로 True(확인)가 반영됨
- return b : False 값으로 선언하지 않으면 취소를 눌러도 기본값이 반영됨
- 강사님이 적은 원본은 함수명을 query() 로 선언, 관례로 추정

- **for Code Newbie**

- 코드를 처음 접하는 비전공자에게는 구조 파악이 상당히 중요하다.
- 그러나 강사와 전공자들에게는 너무 당연시 되어 좀처럼 지적하지 않는 부분, 상세히 설명하지 않는 것이 아래에 해당된다.
- 대체로 자연스럽게 체득되기도 하지만 앞으로 이러한 유형을 정리해야 한다.
- 함수명을 통한 연결

```
<button onclick="f(event)">클릭</button>
<script>
  function f(e){alert(e.type);}
```

- 파라미터를 통한 연결

```
<button onclick="f(event)">클릭</button>
<script>
  function f(e){alert(e.type);}
```

★ 특히, 파라미터 자신을 호출하여 다양한 구문으로 연결된다. (참조변수 this와 유사)

- 차이점
 - 이건 되고

```
<button onclick="f(event)">클릭</button>
<script>
  function f(e){alert(e.type);}
```

- 이건 안된다.

```
<a href="https://www.naver.com" onclick="return a()">네이버 이동</a><hr>
<script>
  function a(e){e.confirm("네이버로 이동하시겠습니까"); return e;}
```

- alert와 confirm은 둘다 내장함수지만, confirm에서는 작동되지 않는다.
- a(){confirm(e.) 이런 구문은 검색도 안된다.
- return e를 a로 바꾸고 이것저것 해봐도 안되더라.

- 그래서 함수의 기능문{ } 안에서 변수를 선언해서 처리한다.

```
function a(){let b = confirm("네이버로 이동하시겠습니까"); return b;}
```

- { let b = 어쩌구저쩌구 }
- 이런 형식이 뒷부분에 많이 나온다.

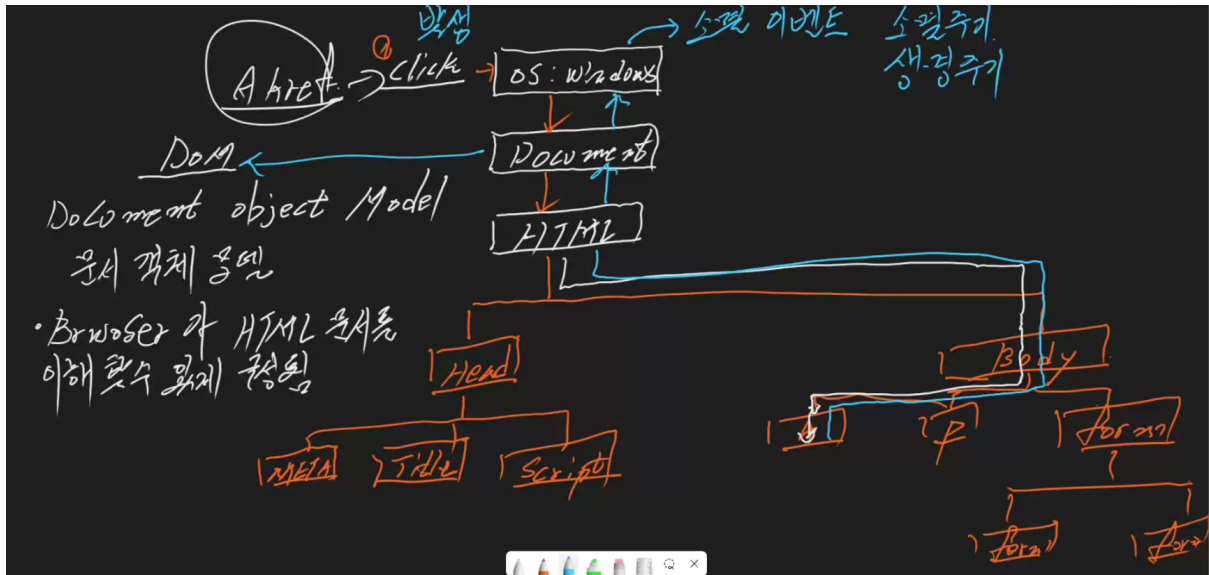
넋두리

- 뭐 자바가 원래 그런거라면 그러려니 하고 넘어가주겠다.
- 물론 코드가 속달되면 여기서 되는 방법을 찾겠지만 그건 지금 단계에선 시간 소모가 크다.
- 미래의 내가 해결해주겠지...
- 주변에 자바스크립트에 능통한 사람이 없으면 학습 효율이 떨어진다.

3] DOM & BOM

□ 보충 필요

1) DOM(Documents Object Model) : 브라우저가 HTML 문서를 이해할 수 있게 구성⁴

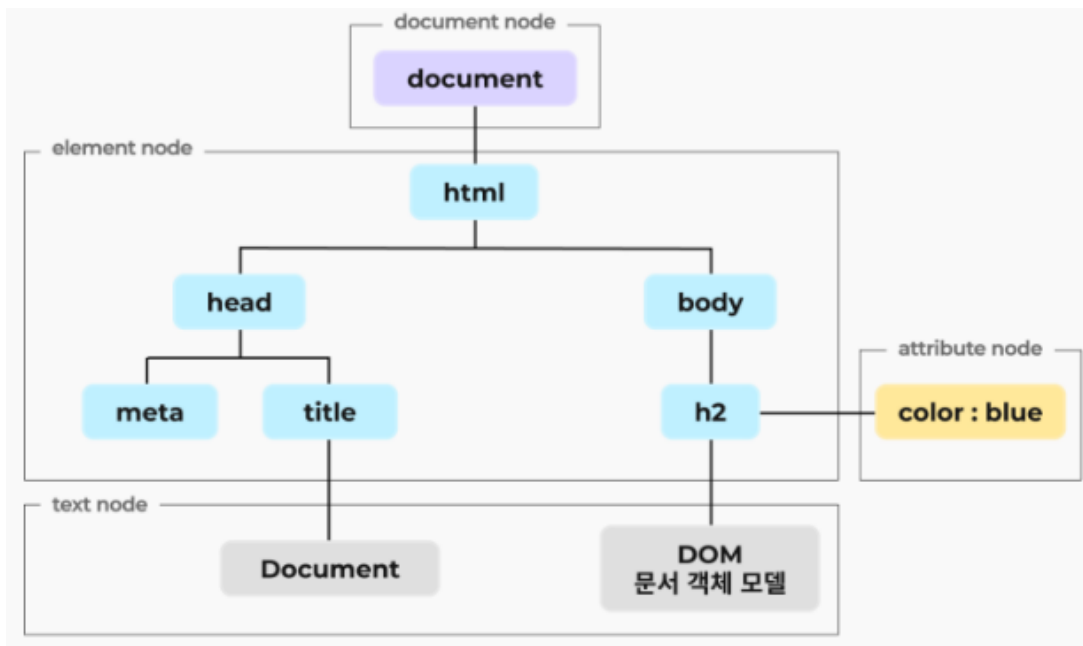


- 윈도우os에서 클릭하여 인풋박스가 생성
 - 웹브라우저가 작동할때 Document를 생성
 - 최초의 문서를 생성하는 것을 DOM
- BOM(Browser Object Model) : 자바스크립트가 주체로 브라우저와의 소통이 목적
 - DOM 생성이후 BOM 생성
 - 자바스크립트의 명령어를 시행하지 말라고?
 - BOM 이 존재하기에 SPA가 가능, 동적처리 가능함을 의미
 - windows : 각 프레임별 하나씩 존재(object)
 - location : URL 정보
 - documents : 현재 문서 정보
 - navigator : 브라우저간 호환성
 - history : 방문 기록
 - screen : 브라우저 외부 환경

⁴ [DOM 소개](#) - Mdn web docs

- DOM 보충 필요⁵

- 정의 : 웹 페이지(HTML이나 XML 문서)의 콘텐츠 및 구조, 그리고 스타일 요소를 구조화시켜 표현하여 프로그래밍 언어가 해당 문서에 접근하여 읽고 조작할 수 있도록 API를 제공하는 일종의 인터페이스
- 필요성 : 정적인 웹이라면 HTML, CSS로도 충분하겠지만 그 이상의 인터랙티브한 기능을 구현하고자 한다면 자바스크립트와 DOM이 필요
- 역할 : 자바스크립트 같은 스크립팅 언어가 쉽게 웹 페이지에 접근하여 조작할 수 있게끔 연결
- 구조 : 웹 페이지, 즉 HTML 문서를 계층적 구조와 정보로 표현하며, 이를 제어할 수 있는 프로퍼티와 메서드를 제공하는 트리 자료구조, HTML DOM, 혹은 HTML DOM Tree로 부르기도 함



- 코딩 파일명 작성법

- **camel** : 첫 문자의 첫 글자는 소문자, 연결된 뒷문자의 첫 글자는 대문자 #myTeam
 - 시작하는 위치를 알 수 있어서 보편적
- **kebab** : 모든 문자는 소문자, 연결될 문자는 -(hyphen)으로 연결 #my-team
- **snake** : 모든 문자는 소문자, 연결될 문자는 _(underbar)으로 연결 #my_team
- **pascal** : 모든 문자의 첫 글자는 대문자 #MyTeam

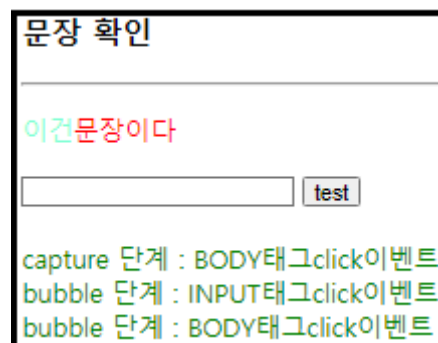
⁵ [문서 객체 모델 DOM 과 자바스크립트 JavaScript | 생성 방식 및 접근 방법](#) - codestate

- Capture & Bubble

```
<body>
  <h3>문장 확인</h3><hr>
  <p style="color : ■aquamarine">이건<span style="color : ■red">문장이다</span></p>
  <form>
    <input type="text">
    <input type="button" value="test" id="button">
  </form>
  <div id="div" style="color : ■green"></div>
  <script>
    let div = document.getElementById("div");
    let button = document.getElementById("button");

    document.body.addEventListener("click", capture, true);
    button.addEventListener("click", bubble, false);
    document.body.addEventListener("click", bubble, false);

    function capture(e){ let obj = e.currentTarget; let tagName = obj.tagName;
      div.innerHTML += "<br> capture 단계 : " + tagName + "태그" + e.type + "이벤트"}
    function bubble(e){ let obj = e.currentTarget; let tagName = obj.tagName;
      div.innerHTML += "<br> bubble 단계 : " + tagName + "태그" + e.type + "이벤트"}
  </script>
</body>
```



버튼 클릭시 각 단계(녹색) 호출

- `let div = document.getElementById("div")` : 이벤트 메시지 출력 공간, div 실행 공간
- `let obj=e.currentTarget` : 이벤트 발생 DOM 안에 기록, 현재 이벤트를 받은 DOM 객체
- `div.innerHTML +=` : 추가, 연속해서 써내려감
- `capture` 단계 : 이벤트를 발생시킨 객체를 찾아 내려가는 단계
- `bubble` 단계 : 이벤트를 발생시킨 객체를 찾고 되돌아가는 단계
- `currentTarget` : 이벤트 리스너가 달린 요소
- `target` : 실제 이벤트가 발생하는 요소

BOM, DOM 은 구조가 그런거니 그러려니 하는데

Capture & Bubble 로 현재가 BOM 과 DOM이 뭔지 알아서 뭐하지?

4] 그외의 다양한 구문들

1) preventDefault : 브라우저의 기본 동작 방지

```
<body>
  <h3>preventDefault</h3><hr>
  <form>
    <input type="checkbox">빵<br>
    <input type="checkbox" onclick="a(event)">밥
  </form>
  <script>
    function a(b){b.preventDefault();}
  </script>
</body>
```

preventDefault

☒ 빵

☐ 밥

밥이 눌러지지 않음

- a href : 링크로 연결 안됨
- checkbox : 체크박스 체크 안됨
- submit : submit 은 적용되나 새로운 창이 실행되지 않음
- 강사님이 적은 원본은 함수명을 **noAction()** 로 선언, 관례로 추정

```
<html>
<script>
  window.onload = function(){
    document.getElementById("checkbox").addEventListener("click", function(event){
      event.preventDefault();
      window.alert('test');
    });
    document.getElementById("anchor").addEventListener("click",function(event){
      event.preventDefault();
      window.alert('test');
    });
    document.getElementById("form").addEventListener("submit", function(event){
      event.preventDefault();
      window.alert('test');
    });
  };
</script>
  <input type="checkbox" id="checkbox"><hr>
  <a href="https://google.com" id="anchor"> google</a><hr>
  <form action="https://google.com" id="form">
    <input type="text" /> <input type="submit" />
  </form>
</html>
```

현직⁶도 잘안쓰는다고 하니 이렇게 있구나 하고 넘어가자

⁶ [\[JavaScript\] preventDefault](#) [란](#), [event.preventDefault](#), [preventDefault\(\)](#)

2) 클릭시 이벤트 객체의 정보를 출력

```
<body>
  <p id="p">클릭하면 이벤트 객체 출력</p>
  <button onclick="f(event)">클릭</button>
  <script>
    function f(e){
      let text = "type : " + e.type + "<br>"
        + "target : " + e.target + "<br>"
        + "currentTarget : " + e.currentTarget + "<br>"
        + "preventDefault : " + e.preventDefault + "<br>"
        + "defaultPrevented : " + e.defaultPrevented;
      let p=document.getElementById("p");
      p.innerHTML = text;
    }
  </script>
</body>
```

```
type : click
target : [object HTMLButtonElement]
currentTarget : [object HTMLButtonElement]
preventDefault : function preventDefault() { [native code] }
defaultPrevented : false
```

클릭

- type : 이벤트 종류, click, load
- target : 이벤트 발생 객체
- currentTarget : DOM
- preventDefault : 디폴트 초기값 취소
- defaultPrevented : 이벤트 취소시의 상태

3) onfocus / onblur

```
<body onload="document.getElementById(name).focus();">
  <form>
    이름 : <input type="text" id="name" onblur="a(this)"><p>
    학번 : <input type="text" id="name">
  </form><br>
</body>
<script>
  function a(b){if(b.value == ""){b.focus();}}
</script>
```

이름 :

학번 :

- onload="document.getElementById(name).focus()" : 없어도됨, 작성이 가능한단 예시
- onfocus : 포커스(마우스 클릭된 경우)상태시 다른 곳으로 이동 불가
- onblur : 포커스 상태 해제

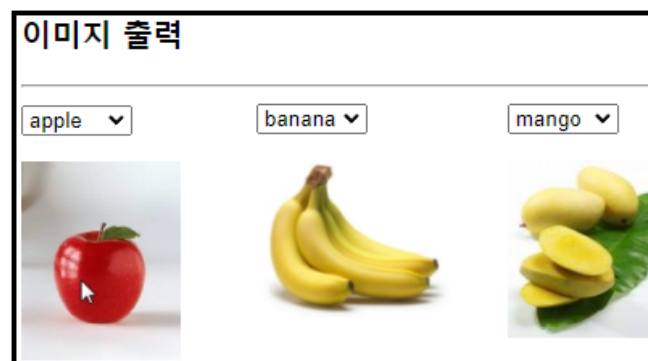
4) eval() : 계산의 결과값을 반환

```
<body>
  <h3></h3><hr>
  계산수식을 입력
  <br><br>
  <form>
    식 : <input type="text" id="exp" value=""><br>
    값 : <input type="text" id="result" value=""><br>
    <input type="button" value="계산" onclick="calculate()">
  </form>
</body>
<script>
  function calculate(){
    let exp = document.getElementById("exp");
    let result = document.getElementById("result");
    result.value = eval(exp.value);
  }
</script>
```

- eval()
 - 문자열로 이루어진 스크립트 코드를 실행
 - 매개변수로 받은 문자 값을 값으로 사용하는 게 아니라 실행문으로 변환해 실행시키기 때문에 문자를 출력하는 게 아니라 문자의 실행 결과를 반환합니다.
- exp.value / result.value
 - 변수명.value : 해당 변수의 값을 받아 처리할 수 있다.
- result.value = eval(exp.value)
 - result.value : result 의 변수 값은 ~ 이다.
 - eval () : () 안의 실행 결과를 반환
 - exp.value : exp 의 변수 값(혹은 수식)
 - eval() 의 수식을 계산하여 result 로 지정
 - 식 : 에 유저가 수식을 작성하면 값 : 으로 계산되어 도출됨

5) SelectImage

```
<body>
  <h3>이미지 출력</h3><hr>
  <form>
    <select id="c" onchange="a()">
      <option value="img\1.png">apple</option>
      <option value="img\2.png">banana</option>
      <option value="img\3.png">mango</option>
    </select>
  </form>
  <p></img></p>
</body>
<script>
  function a(){
    let c = document.getElementById("c");
    let img = document.getElementById("myImg");
    let b = c.selectedIndex
    img.src = c.options[b].value;}
</script>
```




선택 박스에 따라 이미지가 변경

- `<select>` : 선택가능한 박스
 - `<option>` : 박스내의 선택 사항
- `onchange` : 이벤트의(유저의 선택등) 변화를 감지
- `let b = c.selectedIndex` : 유저가 선택한 과일
 - `c` : `select id="c"` 즉, `option` 전체를 의미
 - `selectedIndex` : 선택한 인덱스로
- `img.src = c.options[b].value` : 선택한 과일로 이미지 주소를 변경
 - `img.src` : 이미지의 주소를 변경하라
 - `c.option` : 선택한 과일의
 - `[b].value` : `b` 값으로
 - 현재 변수 `b` 가 `index`를 지칭하므로 `[]` 를 써야함. `()` 쓰면 에러

- 이미지의 크기 출력

```
<body onload="a()">
  <span id="d">이미지 크기</span>
  <p></img></p>
</body>
<script>
  function a(){
    let d = document.getElementById("d");
    let img = document.getElementById("myImg");
    d.innerHTML = img.width + "x" + img.height;
  }
</script>
```

99x124



- onload : 온로드와 함수를 연결해주면 원하는 결과(출력)이 잘된다.
- .innerHTML = img.width , img.height
 - 해당 이미지의 가로세로가 지정한 곳으로 출력된다

- 최종 소스 코드

```
<body onload="changeImage()">
  <h3>이미지 출력</h3><hr>
  <form>
    <select id="sel" onchange="changeImage()">
      <option value="img\1.png">apple</option>
      <option value="img\2.png">banana</option>
      <option value="img\3.png">mango</option>
    </select>
    <span id="mySpan">이미지 크기</span>
  </form>
  <p></img></p>
</body>
<script>
  function changeImage(){
    let sel = document.getElementById("sel");
    let img = document.getElementById("myImg");
    img.onload = function(){
      let mySpan = document.getElementById("mySpan");
      mySpan.innerHTML = img.width + "x" + img.height;
    }
    let index = sel.selectedIndex
    img.src = sel.options[index].value;
  }
</script>
```

- 간소화

```
let index = sel.selectedIndex
img.src = sel.options[index].value

img.src = sel.options[sel.selectedIndex].value
```

6) 라디오 버튼

```
<form>
  <input type="radio" name="city" value="seoul" checked>서울
  <input type="radio" name="city" value="busan">부산
  <input type="radio" name="city" value="suwon">수원
  <input type="button" value="fine checked" onclick="a()">
</form>
<script>
  function a(){
    let b = null;
    let c = document.getElementsByName("city");
    for (let i=0; i<c.length; i++){if(c[i].checked == true) b = c[i];}
    if(b != null){alert(b.value + " 선택됨")}
    else alert("미선택");
  }
</script>
```

☒ 서울
 ☐ 부산
 ☐ 수원

127.0.0.1:5500 내용:
seoul 선택됨

- 태그 내부 조건을 통한 지정

```
<form>
  <select id="c" onchange="a()">
    <option value="img\1.png">apple</option>
  </select>
</form>
</body>
<script>
  function a(){
    let c = document.getElementById("c");
  }
</script>
```

getElementById : select 전체를 id 를 통한 지정

```
<input type="radio" name="city" value="suwon">수원
let c = document.getElementsByName("city");
```

getElementsByName : input 의 name 를 통한 지정

- for (let i=0; i<c.length; i++){ if(c[i].checked == true) b = c[i]}
 - for 문으로 radio 의 값을 하나씩 받아 true 인 곳을 b 값으로 지정
- if(b != null){alert(b.value + " 선택됨")}
 - b 값이 있다면 알림창이 떠서 메시지와 함께 출력