Event

이벤트

- **이벤트**
  - 문서나 브라우저 창에서 특정 순간에 일어난 일

- **리스너(핸들러)**
  - 일어난 이벤트를 적절하게 처리하는 함수(루틴)

- **DOM API에는 200개 가까운 이벤트가 정의되어 있고, 각 브라우저마다 비표준 이벤트를 따로 만들고 있음**

이벤트

예제
https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_onclick

- 이벤트를 등록하는 세 가지 방법
  - inline
  - on* 프로퍼티에 핸들러 등록

  } 이벤트 이름 앞에 " on"이 붙음

  - addEventListener(attachEvent) →선호되는 방식

- **inline**

```
<div onclick="alert('clicked!')">Click me!</div>
```

```
<div id= "div" onclick="handler()">Click me!</div>
<script>
        function handler() {  alert("clicked!"); }
</script>
```

바람직하지 않음
CSS를 inline으로 사용하지 않는 이유와 유사

이벤트

- **on* 프로퍼티에 핸들러 등록**
  - 간단하고 빠른 개발을 위해
  - 모든 요소에는 이벤트 핸들러 프로퍼티가 있으며 onclick처럼 일반적으로 소문자

```
<script>
        var element = document.getElementById("div");
        element.onclick = function () {   alert("clicked!");  }
</script>
```

```
<script>
        var element = document.getElementById(" div ");
        element.onclick = function () {   alert(this.id);  }
</script>//this는 핸들러가 지정된 요소, 여기서는 div요소
```

```
element.onclick =null                    //이벤트 핸들러 제거
```

전역 변수를 사용하는 것과 같으므로 피하는 것이 좋음

이벤트

- **addEventListener**

예제

https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add

target.addEventListener(*type*, *listener*, *options*);

```
<script>
        var element = document.getElementById("div");
        element.addEventListener("click", function () {
                alert("clicked!"); }, false);
</script>
```

on*프로퍼티 이름 → onclick
addEventListener → click

multiple binding

```
<script>
        var element = document.getElementById("div");
        element.addEventListener("click", function () {
                alert(this.id); }, false);
        element.addEventListener("click", function () {
                alert("hello world!"); }, false);
</script>
```

하나의 이벤트에 여러 개의 핸들러를 등록할 수 있음

이벤트

- 이벤트를 등록하는 세 가지 방법
  - 결론: addEventListener를 써라
    - 이벤트의 흐름(구체적으로는 버블링과 캡처링)을 조절할 수 있음
    - 동일한 이벤트에 여러 개의 핸들러를 등록 할 수 있음
    - 표준적인 사용법으로 받아들여지고 있음

이벤트

- event객체
  - DOM과 관련된 이벤트가 발생하면 관련 정보는 모두event라는 객체에 저장
  - 이벤트 핸들러에 매개변수로 전달 가능

```
var element = document.getElementById(" div ");
element.onclick = function (event) {   alert(event.type);  }
```

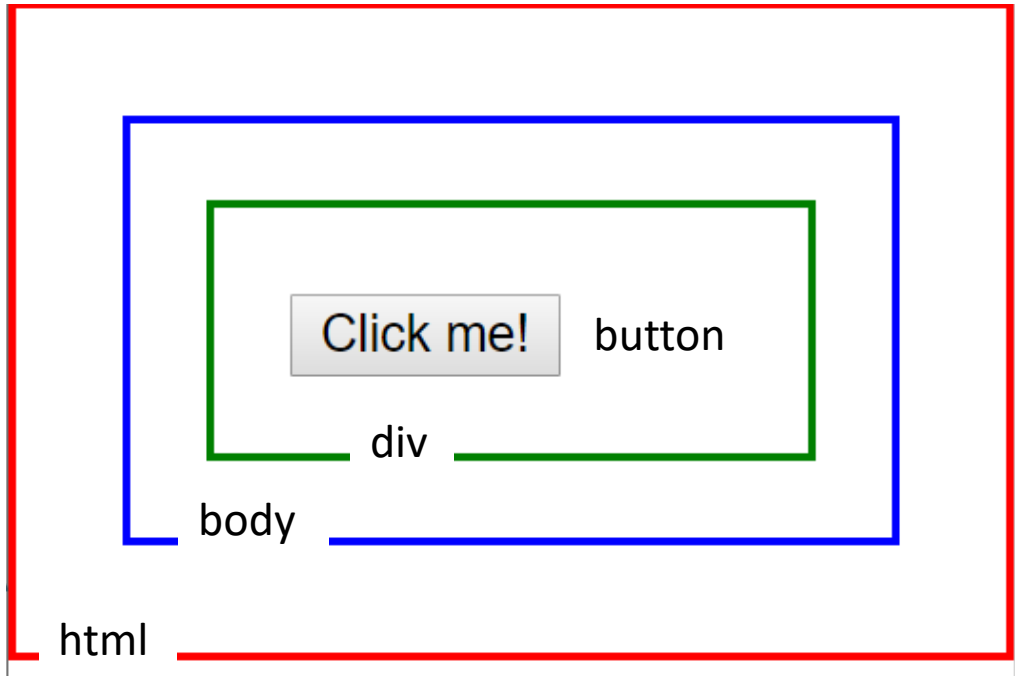
```
var element = document.getElementById("div");
element.addEventListener("click", function (event) {
        alert("event.type"); }, false);
```

이벤트

- event객체
  - 자주 쓰이는 프로퍼티와 메서드
    - target: 발생된 이벤트와 가장 밀접한 요소(클릭의 경우 내가 클릭한 바로 그 요소)
    - type: 발생한 이벤트 타입
    - eventPhase: 1 캡처링 단계, 2 target, 3 버블링 단계
    - preventDefault()
      - 프로그래머가 지정하지 않아도 기본적으로 등록되어 있는 핸들러를 중지
      - cancelable 프로퍼티가 true여야 함
      - 기본 이벤트 예) a태그에 href를 지정하면 핸들러 지정 없이 새로운 웹 페이지로 이동
    - stopPropagation(): 캡처링 혹은 버블링이 더 이상 전파되지 않도록 막음

이벤트

- 이벤트 흐름
  - 버블링(bubbling), 캡처링(capturing)

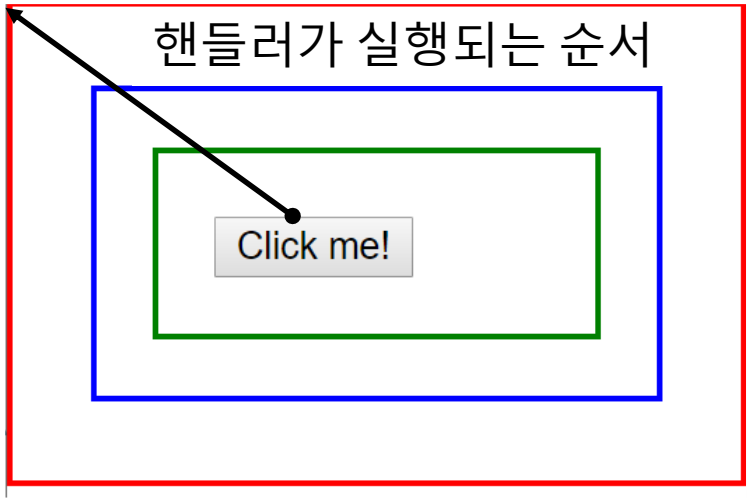


click me!라는 버튼을 클릭하면
버튼을 클릭한 것이지만
문서 전체로 보면
html, body, div를 클릭한 것으로도
볼 수 있음

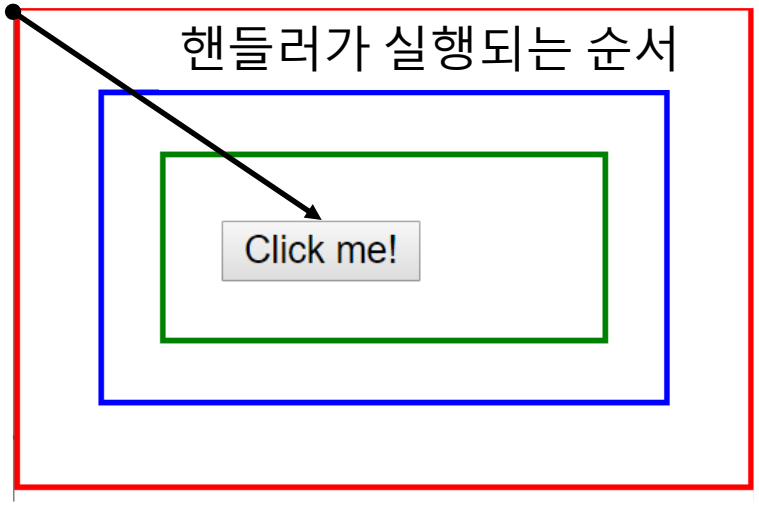
모든 요소에 클릭 이벤트에 대한
핸들러가 등록되어 있다면
어떤 순서(흐름)로 처리 될 것인가?

이벤트

- 이벤트 흐름
  - 버튼을 클릭할 경우, 두 종류의 이벤트 흐름



버블링(bubbling)
이벤트 전파



캡처링(capturing)
이벤트 탐색

target: 발생한 이벤트와 가장 밀접한 관련이 있는 요소
- click me! 버튼을 누르면 target은 button요소
- div를 클릭하면 targe은 div요소

이벤트

https://joshua1988.github.io/web-development/javascript/event-propagation-delegation/

- 이벤트 흐름
  - 버블링

```
<body>
<div class="one">
    <div class="two">
        <div class="three">
        </div>
    </div>
</div>
</body>
```

```
var divs = document.querySelectorAll('div');
divs.forEach(function(div) {
div.addEventListener('click', logEvent); });

function logEvent(event) {
console.log(event.currentTarget.className);
}
```

 클릭 결과

three
two
one

이벤트

- 이벤트 흐름
  - 캡처링

```
<body>
<div class="one">
    <div class="two">
        <div class="three">
        </div>
    </div>
</div>
</body>
```

```
var divs = document.querySelectorAll('div');
divs.forEach(function(div) {
    div.addEventListener('click', logEvent, {
    capture: true // default 값은 false입니다.
    });
});

function logEvent(event) {
    console.log(event.currentTarget.className);
}
```

 클릭 결과

```
three
two
one
```

이벤트

- 이벤트 흐름
  - 흐름 단절: stopPropagation()
  - 버블링이나 캡처링과 같이 다른 요소의 이벤트까지 트리거 되는 것을 막음
  - 앞선 설명과 같이 event객체의 메서드(즉, 핸들러에서 event객체가 필요)
  - https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_event_stoppropagation
  - 위 예제의 func1(event)에서 event는 문자열이 아니라 event객체를 의미

이벤트

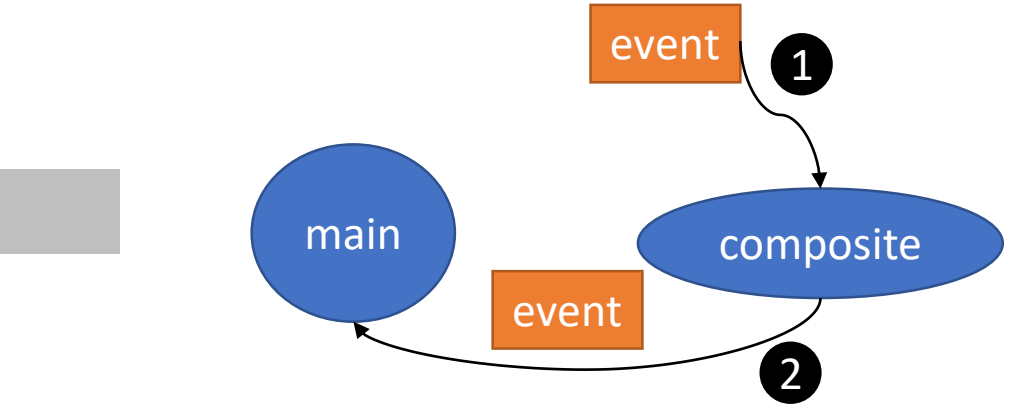
- 이벤트 위임(delegation)
  - 버블링을 이용
  - 요소마다 핸들러를 할당하지 않고, 다수의 요소에 공통으로 적용되는 이벤트 핸들러를 공통된 조상 요소에 단 한 번만 연결
  - 단일 핸들러를 통해 모든 후손 및 미래에 생길 후손(동적 요소)에게까지 이벤트를 적용
    - (예시: https://frontsom.tistory.com/13 , 좋은 예제이므로 이해 바람)
  - 자식요소가 해야 할 이벤트 등록 및 처리를 부모가 대신 → 위임
  - 부모요소는 자식요소에 발생할 이벤트를 감시
  - 이벤트 위임의 장점(정리)
    - 다수의 요소에 공통된 이벤트를 등록할 때 편리
    - 미래에(동적으로) 추가될 요소에도 이벤트 적용 가능

이벤트

- **addEventListener의 옵션**

target.addEventListener(*type*, *listener*, *options*);

```
element.addEventListener('click', doSomething, {
        capture: false,
        once: true,
        passive: false
});
```



- capture(useCapture) – 캡처링 사용 여부 (false가 기본값)
- once - true 면 이벤트가 딱 한번만 발생됨
- passive - true 면 이벤트를 받는 composite 쓰레드가 main 쓰레드의 처리를 기다리지 않고 화면 그리기를 수행→ 즉 이벤트가 발생할 경우 빠르게 화면에 표시(대표적인 예 스크롤)

```
document.addEventListener('touchmove', (e) => {
// Do something...
},
  { passive: true }
);
```

e.preventDefault()는 사용할 수 없음

이벤트

- 자바스크립트 코드의 위치
  - html 안에
    - script태그 안에 자바스크립트 작성
    - script태그의 위치
      - body에 위치할 것인가: html요소가 나오는 body 끝에
      - head에 위치할 것인가: onload사용
  - 외부파일
    - 자바스크립트 코드를 .js파일 안에 넣기
    - .js파일 안에는 script태그를 사용할 필요 없음(CSS파일을 따로 만들듯이)
    - head안에 다음과 같이 src지정

```
<script type="text/javascript"src=" myscript.js"></script>
```

Jquery를 공부하다보면 $(document).ready() vs window.onload의 차이 이해하는 것도 자주 등장

이벤트

종류		설명
마우스 이벤트	onmouseover	지정한 요소에 커서가 올라 갔을 때
	onmouseout	지정한 요소에서 커서가 벗어 났을 때
	onmousemove	지정한 요소 영역에서 커서가 움직일 때
	onclick	지정한 요소를 클릭했을 때
	ondbclick	지정한 요소를 더블클릭 했을 때
키보드 이벤트	onkeypress	지정한 요소에서 키보드가 눌렸을 때(SHIFT, ALT, CTRL, 방향 키 등 기능 키는 제외) 한글X
	onkeydown	모든 키를 눌렀을 때
	onkeyup	키보드를 눌렀다 떼었을 때
기타 이벤트	onfucus	지정한 요소에 포커스가 갔을 때
	onblur	포커스가 왔다가 잃었을 때
	onload	지정한 요소의 하위 요소를 모두 로딩했을 때
	onunload	문서를 닫거나 다른 문서로 이동했을 때
	onsubmit	폼 요소에 전송 버튼을 눌렀을 때
	onreset	폼 요소에 취소 버튼을 눌렀을 때
	onresize	지정된 요소의 크기가 변경되었을 때