JavaScript

DOM Event – 김근형 강사

Dom Event

- Event
  - 해당 프로그램에서 어떤 동작이나 사건이 발생했음 알려주는 신호
  - 이벤트가 일어났을 시 해당 이벤트를 듣는 프로퍼티를 이벤트 리스너(Event Listener)라고 한다
  - 이벤트가 발생할 때 일처리를 할 수 있는 함수나 객체의 메소드를 호출함으로써 동작에 반응할 수 있는 페이지를 만들 수 있고 이벤트가 발생했을 때 일처리를 도와주는 것이 이벤트 핸들러(event handler)라고 한다.

Dom Event

- Event
  - 이벤트 리스너(Event Listener)를 구현 시 두가지 형태로 구현이 가능하다
    - HTML Tag안에 이벤트 속성을 정의
    - 엘리먼트 객체에 Event 프로퍼티 사용
  - 이 두가지 방식이 대표적인 방식이며 항상 이벤트의 이름 앞에 on이 붙는다.

HTML 태그 인라인

```
<button onclick="displayDate()">Try it</button>
```

Event 프로퍼티를 사용

```
document.getElementById("myBtn").onclick = displayDate;
```

Dom Event

- Event Listener 추가
  - addEventListener는 이벤트를 등록하는 가장 권장되는 방식이다. 이 방식을 이용하면 여러개의 이벤트 핸들러를 등록할 수 있다.
  - 이 방식의 중요한 장점은 하나의 이벤트 대상에 복수의 동일 이벤트 타입 리스너를 등록할 수 있다는 점이다.
  - 또한 이벤트 객체를 이용하면 복수의 엘리먼트에 하나의 리스너를 등록해서 재사용할 수 있다.
  - EventListener관련 메서드는 아래와 같다.

메서드 명	설명
addEventListener	해당 객체에 이벤트를 추가한다.
removeEventListener	해당 객체에 추가되어 있던 이벤트를 제거한다.

Dom Event

- addEventListener
  - addEventListener의 형태는 아래와 같다.

```
target.addEventListener(type, listener[, options]);
target.addEventListener(type, listener[, useCapture]);
```

이름	설명
type	반응할 이벤트 유형을 나타내는 대소문자 구분 문자열.
listener	지정된 타입의 이벤트가 발생했을 때, 알림(Event 인터페이스를 구현하는 객체)을 받는 객체이다. EventListener 인터페이스 또는 JavaScript function를 구현하는 객체여야만 한다.
options	이벤트 리스너에 대한 특성을 지정하는 옵션 객체이다. 사용 가능한 옵션은 다음과 같다 **capture**: DOM 트리의 하단에 있는 EventTarget 으로 전송하기 전에, 등록된 listener 로 이 타입의 이벤트의 전송여부를 나타내는 Boolean 이다. **once**: 리스너를 추가한 후 한 번만 호출되어야 함을 나타내는 Boolean입니다. true이면 호출할 때 listener 가 자동으로 삭제된다. **passive**: true일 경우, listener에서 지정한 함수가 preventDefault()를 호출하지 않음을 나타내는 Boolean이다. passive listener 가 preventDefault()를 호출하면 user agent는 콘솔 경고를 생성하는 것 외의 작업은 수행하지 않는다.
useCapture	DOM 트리의 하단에 있는 EventTarget 으로 전송하기 전에, 등록된 listener 로 이 타입의 이벤트의 전송여부를 나타내는 Boolean 이다. 트리에서 위쪽으로 버블링되는 이벤트는 캡처를 사용하도록, 지정된 listener를 트리거하지 않는다. 이벤트 버블링과 캡쳐는 두 요소(엘리먼트)가 해당 이벤트에 대한 핸들(함수)를 등록한 경우, 다른 요소 내에 중첩된 요소에서 발생하는 이벤트를 전파하는 두 가지 방법이다. 이벤트 전파 모드는 요소가 이벤트를 수신하는 순서를 판별한다.

Dom Event

- Event 예제

```
<p id="demo" onclick="myFunction()">Click me.</p>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "YOU CLICKED ME!";
}
</script>
```

```
<p id="demo">Click me.</p>

<script>
document.getElementById("demo").onclick = function() {myFunction()};

function myFunction() {
  document.getElementById("demo").innerHTML = "YOU CLICKED ME!";
}
</script>
```

```
<p id="demo">Click me.</p>

<script>
document.getElementById("demo").addEventListener("click", myFunction);

function myFunction() {
  document.getElementById("demo").innerHTML = "YOU CLICKED ME!";
}
</script>
```

Dom Event

- Event 종류

이벤트	발생
onabort	이미지 로딩에서 이탈하였을 때(다른 연결로 이동)
onactivate	개체의 activeElement 속성을 설정하였을 때
onafterprint	관련된 문서를 인쇄하거나 인쇄 미리보기를 후 즉시
onbeforeactivate	개체의 activeElement 속성이 설정되기 바로 전에
onbeforecopy	선택된 내용이 시스템 클립보드(clipboard)로 복사하기 바로 전에
onbeforecut	선택된 내용이 시스템 클립보드(clipboard)로 잘라내기 바로 전에
onbeforedeactivate	activeElement가 다른 개체로 이동하기 바로 전에
onbeforeeditfocus	편집할 수 있는 용기 개체의 제어가 선택되기 바로 전에
onbeforepaste	시스템 클립보드로부터 데이터를 붙여넣기하기 바로 전에
onbeforeprint	관련된 문서를 인쇄하거나 인쇄 미리보기하기 바로 전에
onbeforeunload	페이지가 언로드되기 바로 전에
onbeforeupdate	연관된 데이터가 업데이트되기 바로 전에
onbegin	이벤트에서 시간이 시작되면 엘레멘트에 발생된다.

Dom Event

- Event 종류

이벤트	발생
onblur	마우스나 탭에 의한 항해로 개체가 초점(포커스)을 상실 했을 때
onbounce	내용이 개체의 한쪽 한계선에 도달하였을 때
oncellchange	데이터 제공자에서 데이터의 내용이 변경되었을 때
onchange	입력폼의 필드 내용이 변경되고 초점을 상실했을 때 발생
onchange(color)	colorpick 비헤이버에서 색상이 변경되면 발생된다.
onclick	왼쪽 마우스가 개체 위를 클릭하였을 때
oncontentready	비헤이버 첨부된 엘레멘트의 내용이 파싱(parse) 완료되면 발생
oncontentsave	비헤이버 첨부된 엘레멘트의 내용이 저장이나 복사되기 전에 발생
oncontextmenu	사용자 지역에서 오른쪽 마우스를 클릭하였을 때
oncontrolselect	사용자가 개체의 제어(control) 선택들 하려고 할 때
oncopy	선택된 내용이 시스템 클립보드(clipboard)로 복사하기 하였을 때
oncut	선택된 내용이 시스템 클립보드(clipboard)로 잘라내기 하였을 때
ondataavailable	데이터 원천 개체로 부터 데이터가 도착하면 주기적으로 발생

Dom Event

- Event 종류

이벤트	발생
ondatasetchanged	데이터 원천 개체의 변경에 의한 감지가 있을 때
ondatasetcomplete	데이터 원천 개체로부터 모든 데이터를 받아 사용할 수 있을 때
ondatasetcomplete	dataselect 비헤이버가 select의 내용을 차지하였을 때 발행
ondblclick	마우스가 개체 위를 두번 클릭하였을 때
ondeactivate	activeElement가 현재의 개체에서 다른 개체로 이동하였을 때
ondetach	엘레멘트에서 첨부된 비헤이버가 제거되기 바로전에 발생
ondocumentready	비헤이버를 포함하는 문서가 파싱(parse)을 완료되었을 때 발생
ondrag	마우스를 눌러 끄는 동안 계속해서 원본 개체에 발생
ondragdrop	NS 화일등의 개체를 창에 드롭다운 시켰을 때
ondragend	마우스를 눌러 끄는 동안 계속해서 원본 개체에 발생
ondragenter	사용자가 드래그하는 개체를 유효한 목표 드롭에서 놓았을 때
ondragleave	드래그하는 마우스를 유효한 목표에서 놓지 않고 이탈했을 때
ondragover	유효한 목표 위에서 드래그하는 동안 연속적으로 발생

Dom Event

- Event 종류

이벤트	발생
ondragstart	개체를 왼쪽 마우스를 누른 상태에서 드래그를 시작하면
ondrop	드래그드롭 작업 중 마우스단추를 놓았을 때
onend	엘레멘트에서 시간이 중지되면 발생
onerror	런타임 오류가 발생하였을 때
onerror	무효한 속성값을 할당하거나 읽기전용에 할당하면 발생
onerror	무효한 속성값을 할당하거나 읽기전용에 할당하면 발생
onerror	속성에 무효한 값을 할당하거나 읽기전용에 할당할 때 발생
onerrorupdate	관련된 데이터가 업데이트되는 동안에 오류가 발생되면 발생
onfilterchange	스타일 필터가 변경되거나 변환을 완료하면 발생
onfinish	마퀴(MARQUEE)의 루프가 완료되면 발생
onfocus	마우스나 탭에 의한 개체에 초점(포커스)이 주어 졌을 때
onfocusin	엘레멘트가 초점을 받았을 때
onfocusout	엘레멘트가 초점을 잃었을 때

Dom Event

- Event 종류

이벤트	발생
onhelp	활성 윈도우에서 도움말을 위하여 F1 키를 누르면 발생
onhide	메디어 플레이어가 감춰지면 발생된다.
onkeydown	키를 개체 위에서 눌렀을 때 발생
onkeypress	키(key)를 개체 위에서 눌렀다 놓았을 때 발생
onkeyup	키를 개체 위에서 놓았을 때 발생
onlayoutcomplete	채워넣기가 끝나고 인쇄하거나 인쇄미리보기를 실행하면 발생
onload	문서를 다시 로딩할 때 원래의 엘레멘트에서 발생
onlosecapture	마우스에 의한 캡쳐(capture)가 상실되었을 때 발생
onmediacomplete	메디어와 연관된 엘레멘트의 로딩이 완료되면 발생
onmediaerror	엘레멘트의 메디어 파일의 로딩이 실패되었을 때 발생
onmedialoadfailed	(불량)엘레멘트의 메디어 파일의 로딩이 실패되었을 때 발생
onmousedown	마우스가 개체 위를 눌렀을 때 발생
onmouseenter	사용자가 마우스 포인터로 개체 위에 들어갔을 때 발생

Dom Event

- Event 종류

이벤트	발생
onmouseleave	마우스포인터가 개체의 범위 밖으로 이동하면 발생
onmousemove	마우스가 개체 위에서 이동하였을 때
onmouseout	마우스가 개체 위에서 이탈하였을 때
onmouseover	마우스가 개체 위로 이동하였을 때
onmouseup	마우스가 개체 위 누른 것을 해제하였을 때
onmousewheel	마우스 굴림단추가 개체 위에서 회전할 때 발생
onmove	사용자나 스크립트로 창의 위치를 이동하였을 때
onmoveend	편집할 수 있는 개체의 이동이 중지되었을 때 발생
onmovestart	개체를 이동하기 시작하면 발생
onopenstatechange	메디어바 플레이어의 열린 상태가 변경될 때 발생
onoutofsync	엘레렌트가 연관된 시간과의 동기성을 상실하면 발생
onpaste	시스템 클립보드(clipboard)로부터 데이터를 붙여넣기하였을 때
onpause	엘레멘트의 시간이 일시중지(pause)하면 발생

Dom Event

- Event 종류

이벤트	발생
onplaystatechange	메디어바 플레이어에서 그 연주 상태가 변경되었을 때 발생
onpropertychange	개체의 속성을 변경하면 그 개체에 발생
onreadystatechange	개체의 상태(state)가 변경되면 발생된다.
onrepeat	시간이 엘레멘트에서 반복되거나 다음번 작동이 시작될 때 발생
onreset	입력폼이 리셋트(reset) 되었을 때
onreset	시간이 begin 값이 되거나 resetElement가 호출되면 발생
onresize	사용자나 스크립트로 창의 크기를 조절하였을 때
onresizeend	사용자가 제어 선택된 개체의 크기 변경을 완료하면 발생
onresizestart	사용자가 제어 선택된 개체의 크기를 변경하기 시작하면 발생
onresume	시간개체가 일시중지에서 다시시작으로 회복되면 발생
onreverse	엘레멘트에서 시간 개체가 뒤로 플레이되면 발생
onrowclick	rowover 비헤이버에서 마우스 커서가 줄을 선택하면 발생
onrowenter	줄이 변경되고 개체에 새로운 값이 있음을 나타내기 위하여 발생

Dom Event

- Event 종류

이벤트	발생
onrowexit	현재 줄을 변경하기의 위한 데이터 원천 제어 바로 전에 발생
onrowout	rowover 비헤이버에서 마우스 커서가 줄에서 나가면 발생
onrowover	rowover 비헤이버에서 마우스 커서가 줄에 들어오면 발생
onrowsdelete	리코드세트(recordset)에서 줄들이 삭제될 상황이 되면 발생
onrowsinserted	현재의 리코드세트에 새로운 줄들이 삽입된 직후에 발생
onsave	웹페이지가 저장, 북마크되거나 다른 페이지로 항해해 가면 발생
onscroll	스크롤되는 개체에서 스크롤 바의 위치를 변경하였을 때 발생
onseek	엘레멘트에서 탐색(seek) 작업이 수행되면 발생
onselect	입력폼에서 입력 필드에서 문자열을 선택(select)하였을 때
onselectionchange	문서의 선택된 부분의 상태가 변경되었을 때 발생
onselectstart	개체가 선택되기 시작하면 발생
onshow	메디어바 플레이어가 보이게 되면 발생된다.
onstart	MARQUEE 개체에서 반복하는 각 루프가 시작될 때 발생

Dom Event

- Event 종류

이벤트	발생
onstop	사용자가 중지 단추를 클릭하거나 웹 페이지를 닫을 때 발생한다
onsubmit	입력폼(form)이 송신(submit) 되었을 때
onsyncrestored	엘레멘트와 그 관련 시간 사이의 동기화가 회복되면 발생
ontimeerror	시간을 지정하는 오류가 일어나면 발생
ontrackchange	ASX 파일에서 정의한 playList에서 트랙이 변경되면 발생된다.
onunload	사용자가 페이지에서 이탈했을 때
onurlflip	+time t:MEDIA 태그에 의해 ASF 파일이 플레이되면 발생

event property

- event property
  - DOM과 관련된 이벤트가 발생하면 관련 정보는 모두 event객체에 저장된다.
  - 이벤트 발생 요소, 이벤트 타입, 이벤트 관련 데이터도 저장된다.
    - ex) 마우스 이벤트 -> 마우스의 위치정보 포함 등, 키보드 이벤트 -> 누른 키의 키코드 등
  - 모든 브라우저가 event 객체를 지원하지만, 세부 사항까지 같지는 않으니 주의할 것.
  - event 객체에는 이벤트 관련 정보와 이벤트를 조작하는 메서드들로 구성된다.

event property

- event property example(1)

```
<input type="text" onkeydown="isKeyPressed(event)">

<p id="demo"></p>

<script>
function isKeyPressed(event) {
  var x = document.getElementById("demo");
  if (event.altKey) {
    x.innerHTML = "The ALT key was pressed!";
  } else {
    x.innerHTML = "The ALT key was NOT pressed!";
  }
}
</script>
```

event property

- event property example(2)

```
<div onmousedown="WhichButton(event)">Click this text with one of your mouse buttons to return a number.
  <p>
  0 = The left mouse button<br>
  1 = The middle mouse button<br>
  2 = The right mouse button
  </p>

  <p><strong>Note:</strong> Internet Explorer 8, and earlier, returns another result:</p>

  <p>
  1 = The left mouse button<br>
  4 = The middle mouse button<br>
  2 = The right mouse button
  </p>
</div>

<script>
function WhichButton(event) {
  alert("You pressed button: " + event.button)
}
</script>
```

event property

- event property example(3)

```
<p>Press a key on the keyboard in the input field to get the Unicode character code of the pressed key.</p>

<input type="text" size="40" onkeypress="myFunction(event)">

<p id="demo"></p>

<p><strong>Note:</strong> The charCode property is not supported in IE8 and earlier versions.</p>

<script>
function myFunction(event) {
  var x = event.charCode;
  document.getElementById("demo").innerHTML = "The Unicode value is: " + x;
}
</script>
```

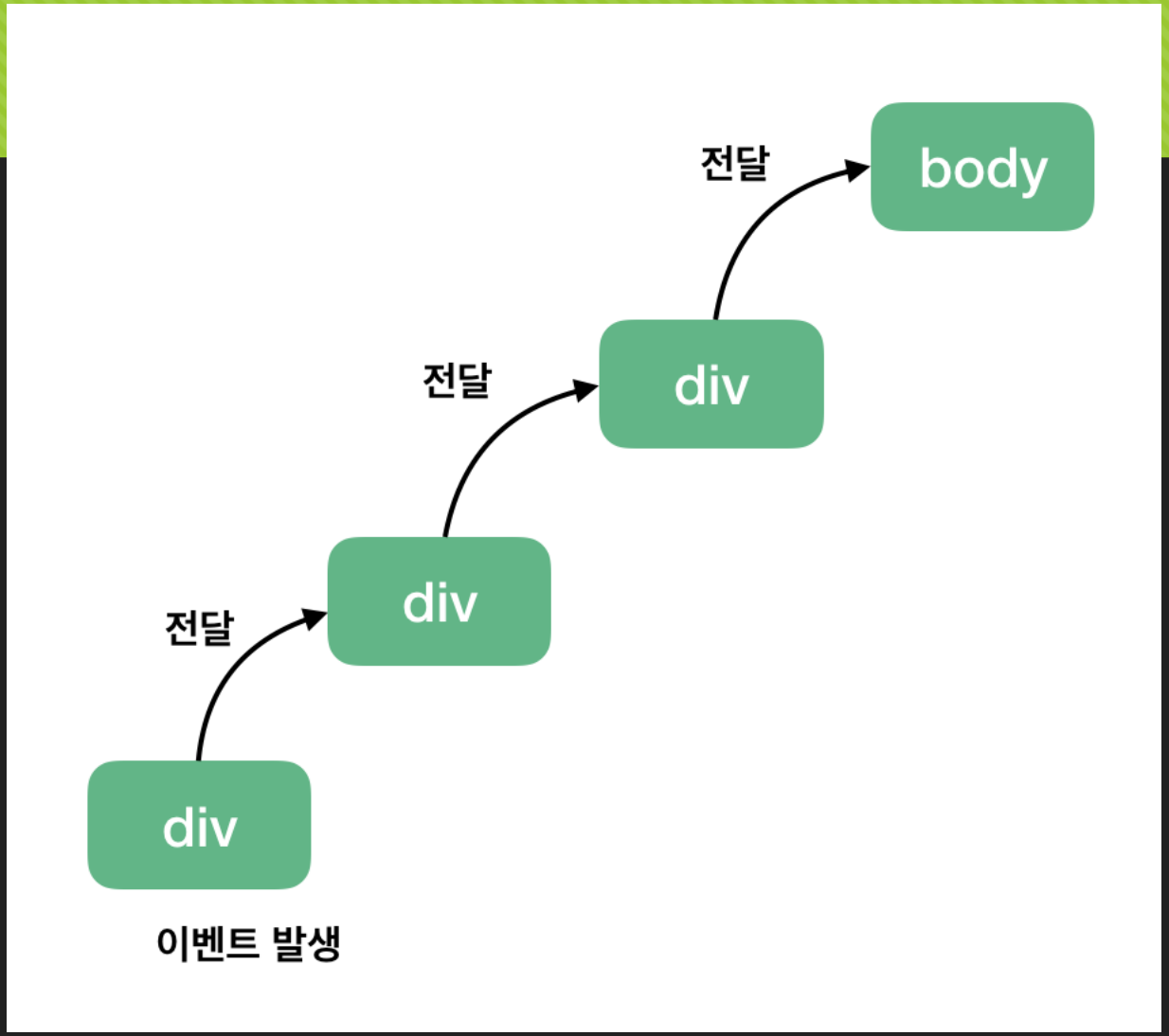
event property

- event property example(4)

```
<h2 onclick="showCoords(event)">Click this heading to get the x (horizontal) and y (vertical) coordinates of the mouse pointer when it was clicked.</h2>

<p><strong>Tip:</strong> Try to click different places in the heading.</p>

<p id="demo"></p>

<script>
function showCoords(event) {
  var x = event.pageX;
  var y = event.pageY;
  var coords = "X coords: " + x + ", Y coords: " + y;
  document.getElementById("demo").innerHTML = coords;
}
</script>
```

Event Bubbling

- 이벤트 버블링(Event Bubbling)
  - 이벤트 버블링은 특정 화면 요소에서 이벤트가 발생했을 때 해당 이벤트가 더 상위의 화면 요소들로 전달되어 가는 특성을 의미한다.



Event Bubbling

- 이벤트 버블링(Event Bubbling)

```
<div class="one">
    <div class="two">
        <div class="three">
        </div>
    </div>
</div>
<script type="text/javascript">
    var divs = document.querySelectorAll('div');
    divs.forEach(function(div) {
        div.addEventListener('click', logEvent);
    });

    function logEvent(event) {
        console.log(event.currentTarget.className);
    }
</script>
```

three

two

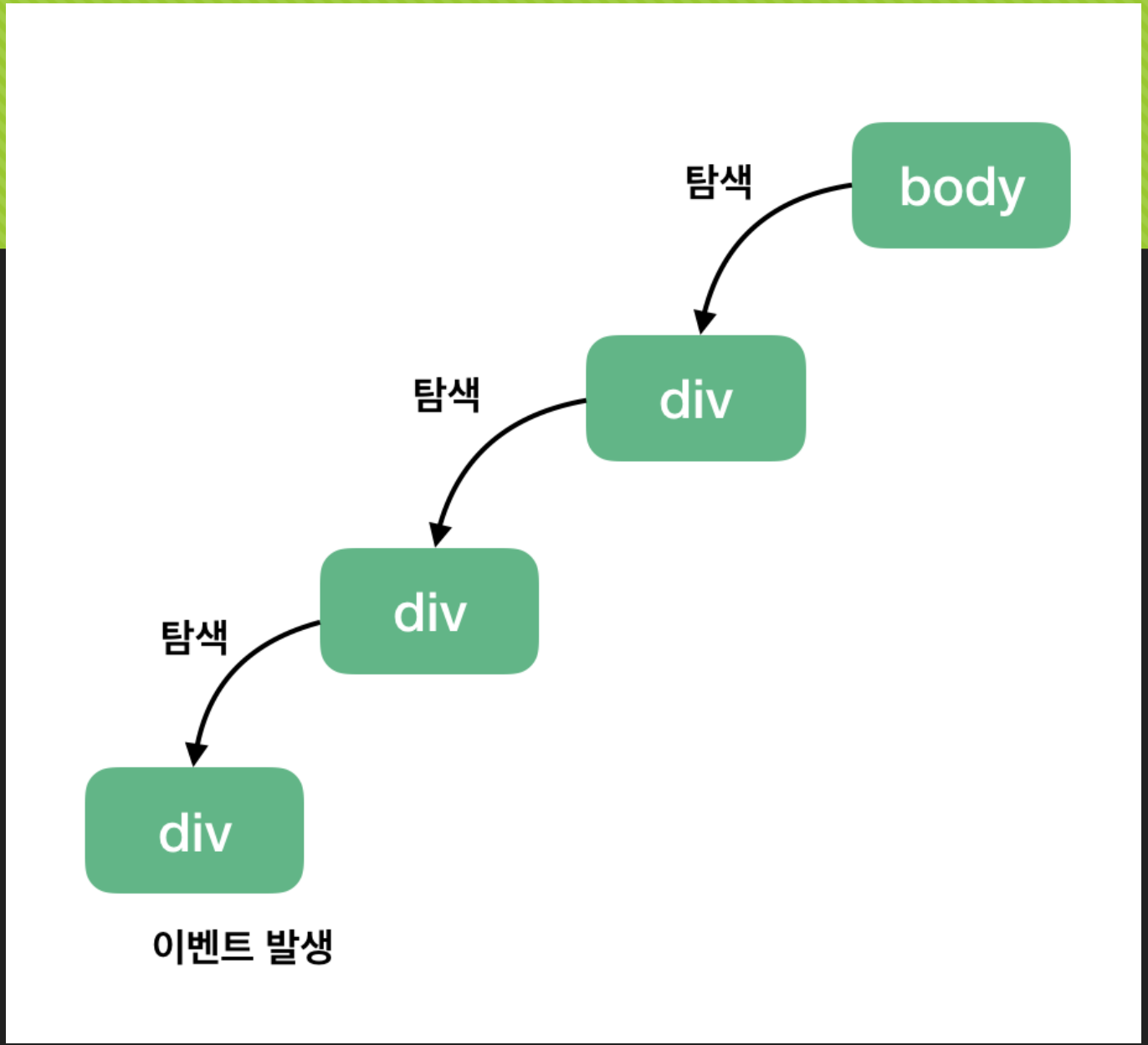
one

Event Bubbling

- 이벤트 버블링(Event Bubbling)
  - 브라우저는 특정 화면 요소에서 이벤트가 발생했을 때 그 이벤트를 최상위에 있는 화면 요소까지 이벤트를 전파시킨다.
  - 따라서, 클래스 명 three -> two -> one 순서로 div 태그에 등록된 이벤트들이 실행된다.
  - 마찬가지로 two 클래스를 갖는 두 번째 태그를 클릭했다면 two -> one 순으로 클릭 이벤트가 동작한다.
  - 여기서 주의해야 할 점은 각 태그마다 이벤트가 등록되어 있기 때문에 상위 요소로 이벤트가 전달되는 것을 확인할 수 있다. 만약 이벤트가 특정 div 태그에만 달려 있다면 위와 같은 동작 결과는 확인할 수 없다.
  - 이와 같은 하위에서 상위 요소로의 이벤트 전파 방식을 이벤트 버블링(Event Bubbling)이라고 한다.

Event Capture

- 이벤트 캡쳐(Event Capture)
  - 이벤트 캡쳐는 이벤트 버블링과 반대 방향으로 진행되는 이벤트 전파 방식이다.
  - addEventListener() API에서 옵션 객체에 capture:true를 설정해주면 해당 이벤트를 감지하기 위해 이벤트 버블링과 반대 방향으로 탐색한다.



Event Capture

- Event Capture

```
<div class="one">
    <div class="two">
        <div class="three">
        </div>
    </div>
</div>
<script type="text/javascript">
    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);
    }
</script>
```

one

two

three

event.stopPropagation

- event.stopPropagation
  - event.stopPropagation은 이벤트가 전파되는 것을 막는다.

```
<script type="text/javascript">
    var divs = document.querySelectorAll('div');
    divs.forEach(function(div) {
        div.addEventListener('click', logEvent);
    });

    function logEvent(event) {
        console.log(event.currentTarget.className)
        event.stopPropagation();
    }
</script>
```

```
<script type="text/javascript">
    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);
        event.stopPropagation();
    }
</script>
```

event.stopPropagation

- 그 외 이벤트를 막는 메서드

메서드	내용
event.preventDefault	a 태그의 링크 이동, form 태그의 전송을 막아주는 역할을 한다.
event.stopImmediatePropagation	부모에게는 어떠한 이벤트도 버블링되지 않으면서 다른 클릭 이벤트도 실행되지 않는다.

Event Delegation(이벤트 위임)

- Event Delegation(이벤트 위임)
  - 하위 요소에 각각 이벤트를 붙이지 않고 상위 요소에서 하위 요소의 이벤트들을 제어하는 방식이다.
  - 이 방법은 위의 이벤트 버블링, 이벤트 캡쳐를 이용해서 작성할 수 있는 이벤트 제어 방식이다.

```
<h1>오늘의 할 일</h1>
<ul class="itemList">
    <li>
        <input type="checkbox" id="item1">
        <label for="item1">이벤트 버블링 학습</label>
    </li>
    <li>
        <input type="checkbox" id="item2">
        <label for="item2">이벤트 캡쳐 학습</label>
    </li>
</ul>
<script type="text/javascript">
    // var inputs = document.querySelectorAll('input');
    // inputs.forEach(function(input) {
    //  input.addEventListener('click', function() {
    //      alert('clicked');
    //  });
    // });
    var itemList = document.querySelector('.itemList');
    itemList.addEventListener('click', function(event) {
        alert('clicked');
    });
```

DOMContentLoaded 이벤트

- DOMContentLoaded 이벤트
  - 보통 DOM 트리에 접근하기 위해 load 이벤트 발생을 기다린 다음에 스크립트를 실행한다.
  - 하지만 load 이벤트는 HTML 파일 이외의 외부 리소스까지 전부 로드 되어야 접근이 가능하도록 설계되기 때문에 스크립트의 실행이 매우 느리다.
  - DOMContentLoaded 이벤트는 HTML과 스크립트를 로드한 후 브라우저 내부에서 DOM이 구성된 시점에서 발생한다.
  - 즉, 본문에 포함된 이미지나 CSS에서 부르는 이미지 등의 로드를 기다리지 않고 DOMContentLoaded 이벤트가 발생한다.
  - JQuery에서 쓰는 $(document).read(function(){})도 DOMContentLoaded 이다.

DOMContentLoaded 이벤트

- DOMContentLoaded 이벤트 사용방법

```
window.addEventListener('DOMContentLoaded', function(){});
```

```
<body>
    <script type="text/javascript">
        window.addEventListener('DOMContentLoaded', (event) => {
            alert('DOM fully loaded and parsed');
        });
    </script>
</body>
```

DOM 트리의 생성 시점

- DOM 트리의 생성 시점
  - DOM 트리 생성은 html 문서를 다 읽은 뒤에 생성이 이루어진다.
  - 만약 그 전에 자바스크립트를 통해 DOM 트리에 접근하려고 했다면 해당 함수에서는 null을 반환한다.
  - DOM 트리가 생성되기 이전 시점에 자바스크립트를 접근하려고 했기 때문이다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script type="text/javascript">
        var t = document.getElementById('target');
        console.log(t);
    </script>
</head>
<body>
    <p id="target">Hello</p>
</body>
</html>
```

DOM 트리의 생성 시점

- DOM 트리의 생성 시점
  - 이것을 해결하기 위해 위에서 이야기한 Load와 DOMContentLoaded 가 필요하다.
  - load는 모든 컨텐츠가 로드 된 시점에서 쓰고자 할 경우 쓸 수가 있으며 DOMContentLoaded는 모든 컨텐츠의 로드 없이 HTML을 통해 DOM Tree가 생성 완료된 시점이면 바로 가능하다.
  - 속도는 load보단 DOMContentLoaded가 더 빠르다

```
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script type="text/javascript">
        window.addEventListener('load', function(){
            var t = document.getElementById('target');
            console.log(t);
        })
    </script>
</head>
<body>
    <p id="target">Hello</p>
</body>
```

```
<p id="target">Hello</p>
```

```
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script type="text/javascript">
        window.addEventListener('DOMContentLoaded', function(){
            var t = document.getElementById('target');
            console.log(t);
        })
    </script>
</head>
<body>
    <p id="target">Hello</p>
</body>
```

```
<p id="target">Hello</p>
```