

삼성 청년 SW 아카데미

JavaScript

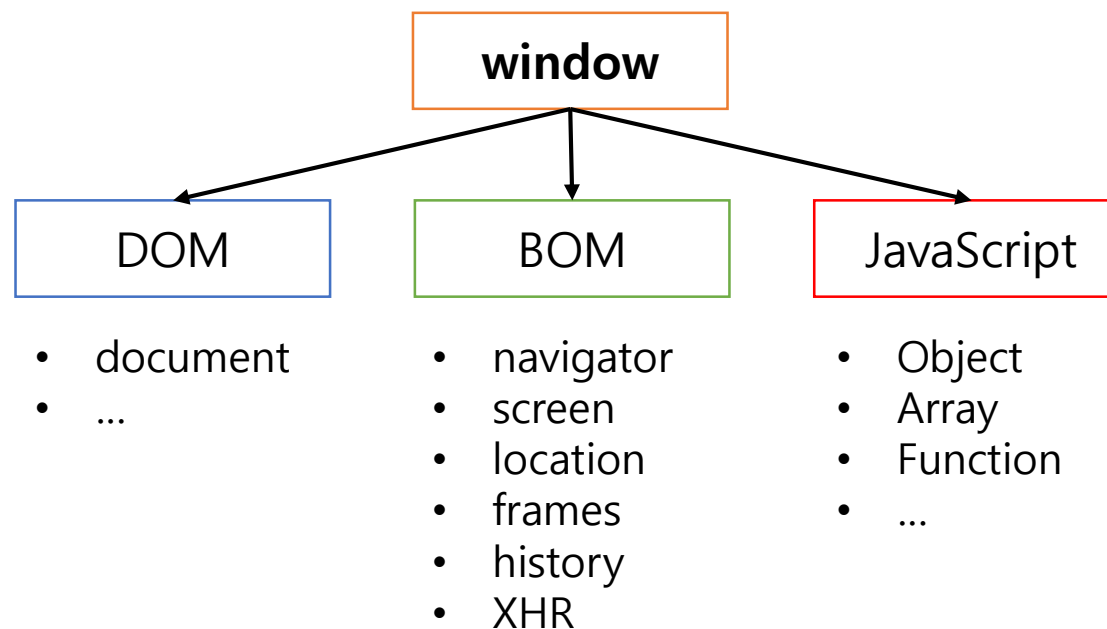
JavaScript

- DOM
- EVENT

DOM

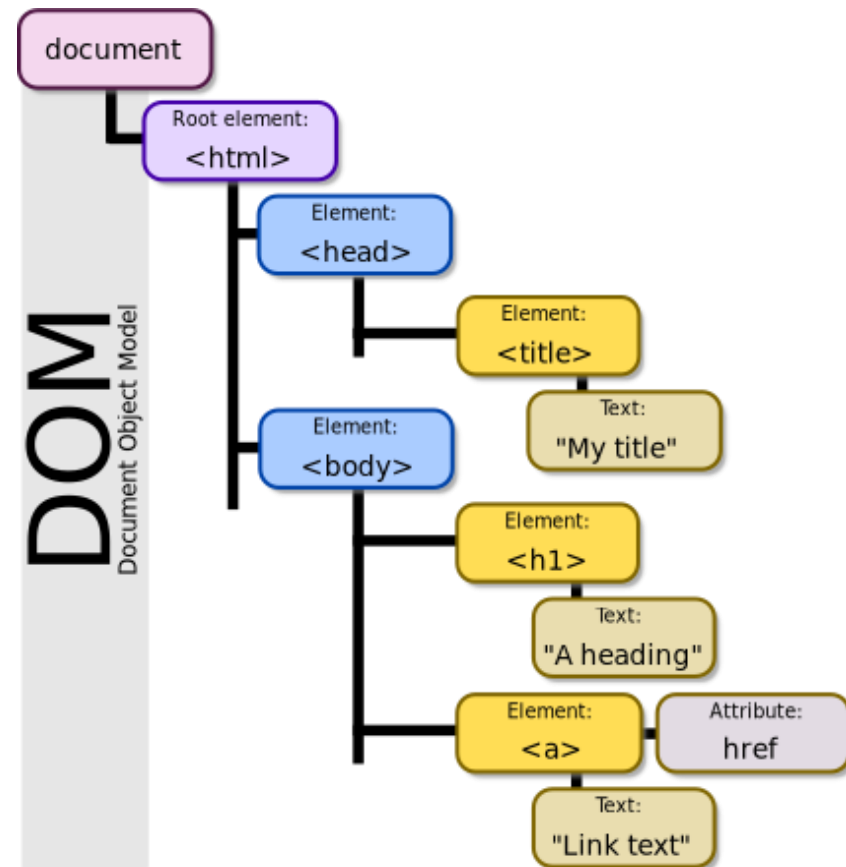
✓ window 제공 함수

- alert
- confirm
- prompt
- open
- parseInt, parseFloat
- setTimeout, clearTimeout
- setInterval, clearInterval



✓ DOM

- XML, HTML 문서의 각 항목을 계층으로 표현하여 생성, 변형, 삭제할 수 있도록 돕는 인터페이스
- DOM은 문서 요소 집합을 트리 형태의 계층 구조로 HTML 표현
- HTML 문서의 요소를 제어하기 위해 지원
- 상단의 document 노드를 통해 접근



✓ 문서 접근 방식 이해

- getElementById(string)
- querySelector(css selector)
- querySelecotrAll(css selector)

✓ getElementById(string)

```
var ele = document.getElementById("a");
```

<section>

<h2>테스트</h2>

<div id="a" class="b">지역</div>

<div class="b">광주</div>

<div name="c">구미</div>

<p name="c">대전</p>

<p class="d">서울</p>

<p class="e">부울경</p>

</section>

- ✓ querySelector(css selector)

```
var ele = document.querySelector("#a");  
var ele = document.getElementById("a");
```

<section>

<h2>테스트</h2>

<div id="a" class="b">지역</div>

<div class="b">광주</div>

<div name="c">구미</div>

<p name="c">대전</p>

<p class="d">서울</p>

<p class="e">부울경</p>

</section>

- ✓ querySelector(css selector)

```
var ele = document.querySelector("div");
```

<section>

<h2>테스트</h2>

<div id="a" class="b">지역</div>

<div class="b">광주</div>

<div name="c">구미</div>

<p name="c">대전</p>

<p class="d">서울</p>

<p class="e">부울경</p>

</section>

- ✓ querySelector(css selector)

```
var ele = document.querySelector(".b");
```

<section>

<h2>테스트</h2>

<div id="a" class="b">지역</div>

<div class="b">광주</div>

<div name="c">구미</div>

<p name="c">대전</p>

<p class="d">서울</p>

<p class="e">부울경</p>

</section>

- ✓ querySelector(css selector)

```
var ele = document.querySelector("[name='c']");
```

<section>

<h2>테스트</h2>

<div id="a" class="b">지역</div>

<div class="b">광주</div>

<div name="c">구미</div>

<p name="c">대전</p>

<p class="d">서울</p>

<p class="e">부울경</p>

</section>

✓ querySelectorAll(css selector)

querySelector와 사용방식 동일

결과를 배열처럼 사용

```
var list = document.querySelectorAll("div");
```

```
for (var i = 0; i < list.length; i++) {
```

```
    console.log( list[i] )
```

```
}
```

- ✓ querySelectorAll(css selector)

```
var ele = document.querySelector("#a");
```

<section>

<h2>테스트</h2>

<div id="a" class="b">지역</div>

<div class="b">광주</div>

<div name="c">구미</div>

<p name="c">대전</p>

<p class="d">서울</p>

<p class="e">부울경</p>

</section>

- ✓ querySelectorAll(css selector)

```
var ele = document.querySelectorAll("div");
```

<section>

<h2>테스트</h2>

<div id="a" class="b">지역</div>

<div class="b">광주</div>

<div name="c">구미</div>

<p name="c">대전</p>

<p class="d">서울</p>

<p class="e">부울경</p>

</section>

- ✓ querySelectorAll(css selector)

```
var ele = document.querySelectorAll(".b");
```

<section>

<h2>테스트</h2>

<div id="a" class="b">지역</div>

<div class="b">광주</div>

<div name="c">구미</div>

<p name="c">대전</p>

<p class="d">서울</p>

<p class="e">부울경</p>

</section>

- ✓ querySelectorAll(css selector)

```
var ele = document.querySelectorAll("[name='c']");
```

<section>

<h2>테스트</h2>

<div id="a" class="b">지역</div>

<div class="b">광주</div>

<div name="c">구미</div>

<p name="c">대전</p>

<p class="d">서울</p>

<p class="e">부울경</p>

</section>

✓ 문서 조작 방식 이해

- createElement(tagName)
- createTextNode(text)
- appendChild(node)
- append(string | node)
- removeChild(node)
- setAttribute(name, value)
- innerHTML
- innerTEXT

- ✓ createElement(string), append(string | node)

엘리먼트 생성

```
var ele = document.createElement("img");
```

추가할 기존 엘리먼트 접근

```
var parent = document.getElementById("list");
```

엘리먼트 추가

```
parent.append(ele);
```

```
<div id="list"></div>
```

메모리에 생성
(화면에 보이지 않는 상태)

```
<img>
```

```
<div id="list">  
  <img>  
</div>
```

✓ setAttribute(name, value)

```
var ele = document.createElement("img");
```

생성된 img 엘리먼트에 속성 추가하기

```
ele.setAttribute("src", "./images/cake.jpg");
```

```
ele.setAttribute("width", 200);
```

```
ele.setAttribute("height", 150);
```

```
ele.src = "./images/cake.jpg";
```

```
ele.width = 200;
```

```
ele.height = 150;
```

```
<img>
```

```

```

✓ 속성 설정 시 주의점 - 사용자 정의 속성

```
var ele = document.createElement("img");  
ele.setAttribute("src", "./images/cake.jpg");  
ele.setAttribute("width", 200);  
ele.setAttribute("height", 150);  
ele.setAttribute("msg", "test");
```

```
ele.src = "./images/cake.jpg";  
ele.width = 200;  
ele.height = 150;  
ele.msg = "test"
```

```

```

```

```

✓ innerHTML을 이용한 요소내용 변경

조작할 엘리먼트 접근

```
var list = document.getElementById("list");
```

엘리먼트의 innerHTML 을 접근

```
list.innerHTML
```

처리할 작업 진행

```
list.innerHTML = "<img src='./images/cake.jpg'  
width='200' height='150' />";
```

```
<div id="list">
```



```
</div>
```

```
<div id="list">
```

```
  <img src='./images/cake.jpg'  
    width='200' height='150' />
```

```
</div>
```

이벤트 - event

✓ Event

- (특히 중요한) 사건[일]
- 웹 페이지에서 여러 종류의 상호작용이 있을 때 마다 이벤트가 발생
- 마우스를 이용했을 때, 키보드를 눌렀을 때 등 많은 이벤트가 존재
- JavaScript를 사용하여 DOM에서 발생하는 이벤트를 감지하고 대응하는 작업을 수행할 수 있음

✓ 이벤트 종류

- 키보드 → keyup, keydown, keypress
- 마우스 → click, mousemove, mouseup, mousedown, mouseenter, mouseleave
- 로딩 → load, unload
- 폼 → input, change, blur, focus, submit

✓ 이벤트 처리 방식의 이해

- 고전 이벤트 처리 방식: attribute / property 방식으로 등록
- 표준 이벤트 처리 방식: addEventListener 메서드 이용

✓ 고전 이벤트 처리 방식 - 1

- 인라인 이벤트 설정 → 엘리먼트에 직접 지정
- 설정하려는 이벤트를 정하고 **on이벤트종류** 의 형식으로 지정

버튼을 클릭 했을 때 경고창을 띄우자

```
<button onclick="alert('click')">클릭</button>
```

실행할 코드가 많다면 함수를 만들고 함수를 호출하자.

```
<button onclick="doAction();">클릭</button>
```

```
function doAction() {  
  var sum = 0;  
  for (var i = 0; i < 10; i++) {  
    sum += i;  
  }  
  alert( sum );  
}
```

✓ 고전 이벤트 처리 방식 - 2

- 엘리먼트에서 이벤트를 직접 설정하지 않고 스크립트에서 이벤트 설정

```
var btn = document.querySelector("#btn");
```

```
<button id="btn">버튼</button>
```

```
btn.onclick = doAction;
```

```
function doAction() {
```

```
    alert("클릭");
```

```
}
```

✓ 표준 이벤트 처리 방식 - 3

- 이벤트 요소.addEventListener(이벤트타입, 이벤트리스너, [option]);

```
var btn = document.querySelector("#btn");           <button id="btn">버튼</button>
```

```
btn.addEventListener("click", doAction );
```

```
function doAction() {
```

```
    alert("클릭");
```

```
}
```

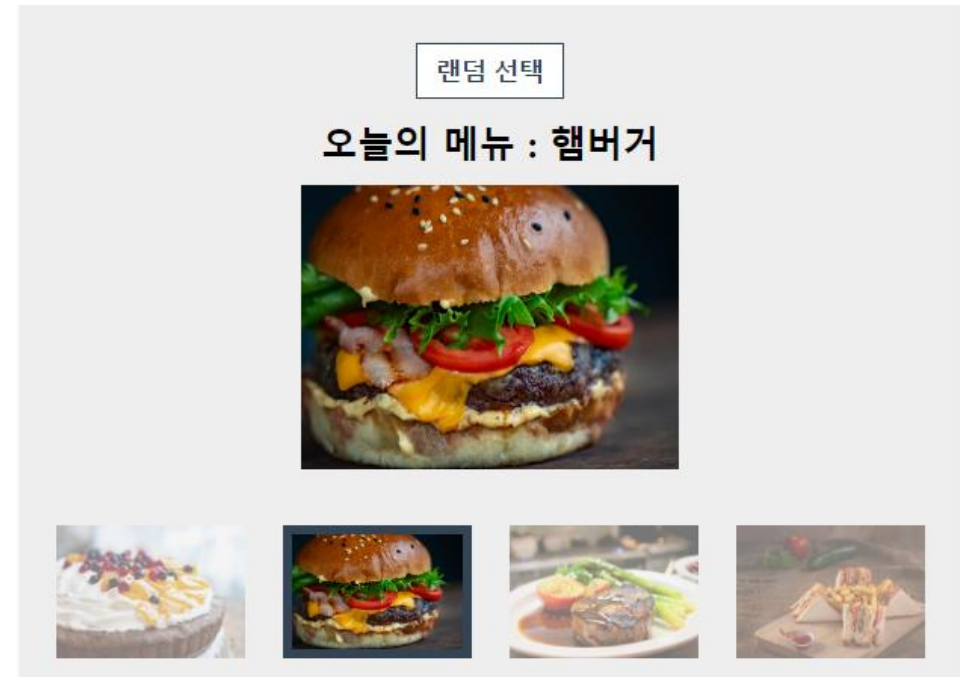
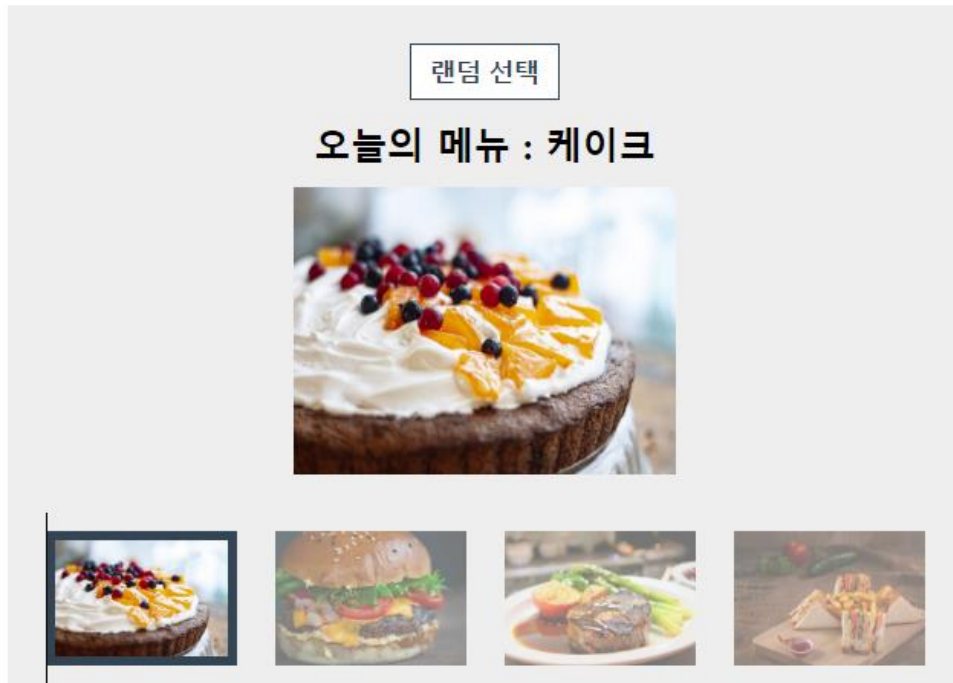
✓ 이벤트 전파(Event propagation)

- 캡처링 단계(capturing phase) : 이벤트가 상위 요소에서 하위 요소 방향으로 전파
- 타깃 단계(target phase): 이벤트가 타깃에 도달
- 버블링 단계(bubbling phase): 이벤트가 하위 요소에서 상위 요소 방향으로 전파

✓ 고전처리방식 vs 표준 처리 방식

- 고전 처리방식: 타깃 단계와 버블링 단계의 이벤트만 캐치 가능
- 표준 처리방식: 타깃 단계와 버블링 단계 뿐만 아니라 캡처링 단계의 이벤트도 선별적으로 캐치 가능
- 캡처링 단계의 이벤트를 캐치하려면 addEventListener의 3번째 인수로 true를 전달
- 3번째 인수를 생략하거나, false를 전달 => 타깃단계와 버블링 단계의 이벤트 캐치

✓ 실습 (메뉴 랜덤 선택)



다음 방송에서 만나요!

삼성 청년 SW 아카데미