

08

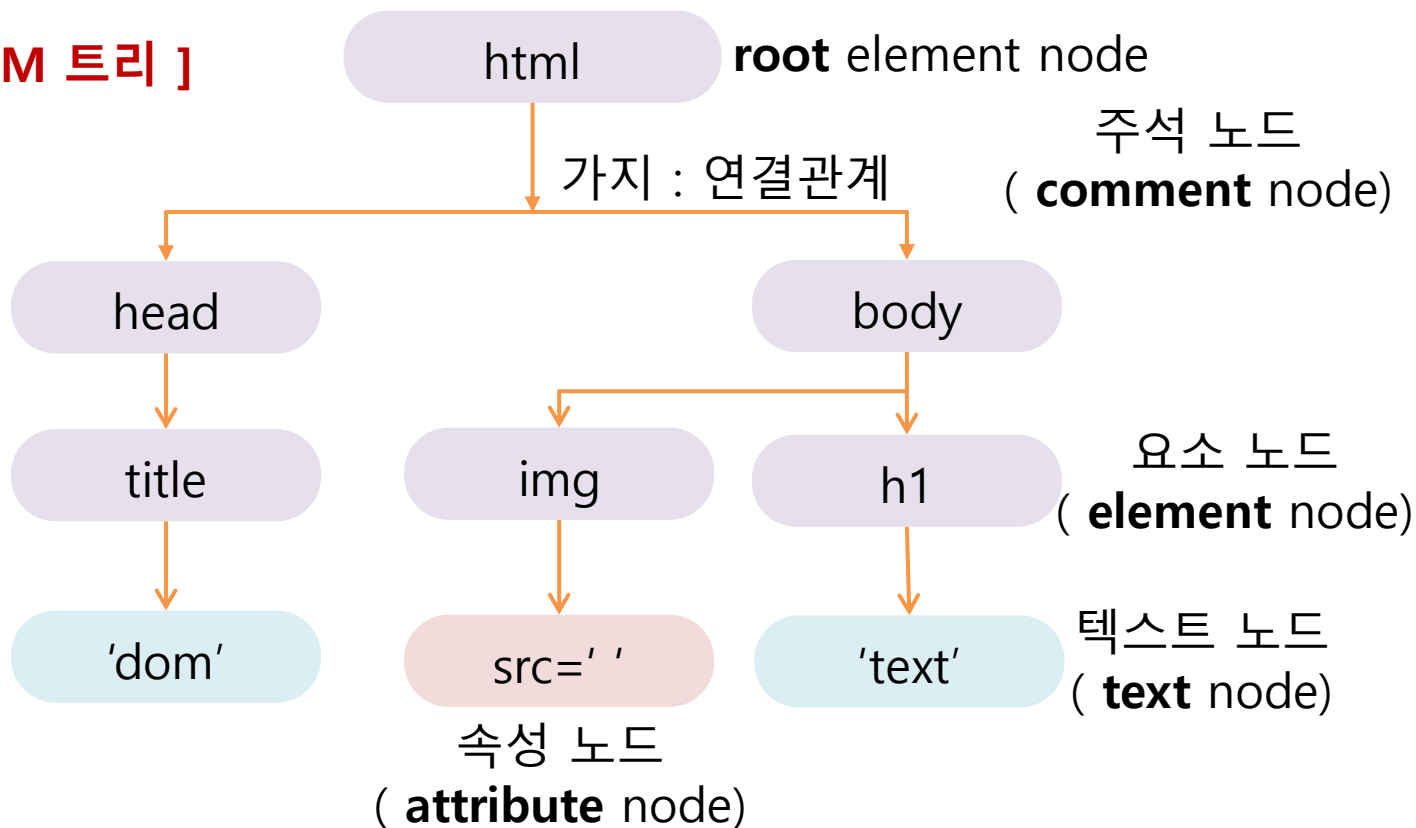
문서 객체 모델(DOM)

- DOM 객체

❖ document 객체(DOM)

- 웹 문서에 접근하고 제어할 수 있도록, 문서를 체계적으로 정리한 객체 집합
- HTML 페이지 읽으면서 문서 객체 생성(정적)
- Javascript 로 문서 객체 생성(동적)

[DOM 트리]



❖ document 객체(DOM)

문서 객체 가져오기

| 메서드 | 설명 |
|--|-------------------|
| <code>getElementById(id)</code> | Id가 일치하는 요소 |
| <code>getElementsByClassName(class)</code> | class가 일치하는 요소들 |
| <code>getElementsByTagName(tagName)</code> | tagName이 일치하는 요소들 |
| <code>getElementsByName(name)</code> | name이 일치하는 요소들 |
| <code>querySelector('선택자')</code> | 선택자로 첫번째 요소 |
| <code>querySelectorAll('선택자')</code> | 선택자와 일치하는 모든 요소 |
| <ul style="list-style-type: none"> - 직접 선택자 : id, class, formName, elementName - 간접 선택자 : parentNode, childNode, firstChild, children, nextSibling, previousSibling, | |

❖ document 객체(DOM)

웹 요소 가져오기

- `querySelector()` : 하나의 요소만 가져옴
 - 여러 개인 경우 첫번째 요소만 가져옴
- `querySelectorAll()` : 여러 개의 요소를 배열 형태로 가져옴
 - `nodeList()`에 저장

웹 요소 내용 가져오기

- 웹 요소.`innerText` – 브라우저 창에 보이는 내용만 가져옴
- 웹 요소.`innerHTML` – 태그와 함께 가져옴
- 웹 요소.`textContent` – 소스에 입력되어진 그대로 가져옴

❖ document 객체(DOM)

웹 요소 내용 수정하기

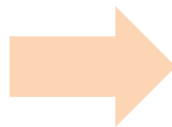
- 웹 요소.innerText = 내용
- 웹 요소.innerHTML = 내용
- 웹 요소.textContent = 내용
- 이미지 요소.src = 이미지 파일
- 웹 요소.style.속성명 = 속성값
 - 두 단어 이상 속성명 :
background-color => backgroundColor

❖ document 객체(DOM)

문서 객체 속성 변경

```
<h1 id="heading"> 문서 객체 속성 변경 </h1>
```

문서 객체 속성 변경



이미지 변경



```
img src="coffee-pink.jpg" id="cup" width="200" height="200">
```

❖ document 객체(DOM)

문서 객체 속성 변경

```
<h1 id="heading">문서 객체 속성 변경</h1>
```

```

```

```
<script>
```

```
document.querySelector('#heading').innerHTML = '이미지 변경';
```

```
let bigPic = document.querySelector("#cup");
```

```
console.log(bigPic.getAttribute('src')); => img/coffee-pink.jpg
```

```
bigPic.setAttribute('src', 'coffee-gray.jpg');
```

```
</script>
```

❖ document 객체(DOM)

문서 객체 스타일 변경

```
let header = document.getElementById('heading')  
header.style.border = '2px solid red';  
header.style.backgroundColor = 'pink';
```

문서 객체 제거

```
let cup = document.getElementById('cup');  
cup.parentNode.removeChild(cup); //부모 노드를 이용한 삭제  
cup.remove() // 바로삭제
```


❖ 이벤트(Event)

이벤트 처리

- 이벤트가 발생하면 바로 이벤트 처리 함수 연결
- 이벤트 발생은 이벤트 이름 앞에 '**on**' 을 붙여 사용
- 이벤트 처리기 = 실행명령이나 함수 연결
- 태그에 직접 입력하는 **인라인 이벤트 모델**

```
<h1 id="heading" onclick="alert('click ')">이벤트 처리</h1>
```

- **DOM을 이용한 이벤트 연결**

```
document.getElementById('heading').onclick=function(){  
    alert('click');  
}  
  
document.getElementById('heading').onclick= null --- 제거
```

❖ 이벤트(Event)

표준 이벤트 처리

- 이벤트 연결과 제거

addEventListener(eventName, handler, **useCapture**)

removeEventListener(eventName, handler)

```
let header = document.getElementById('heading')
let handler = function(){
  this.style.color = 'red'; --- this : 이벤트 발생 객체
  alert('click');
}
header.addEventListener('click', handler); --- 연결
header.removeEventListener('click', handler); --- 제거
```

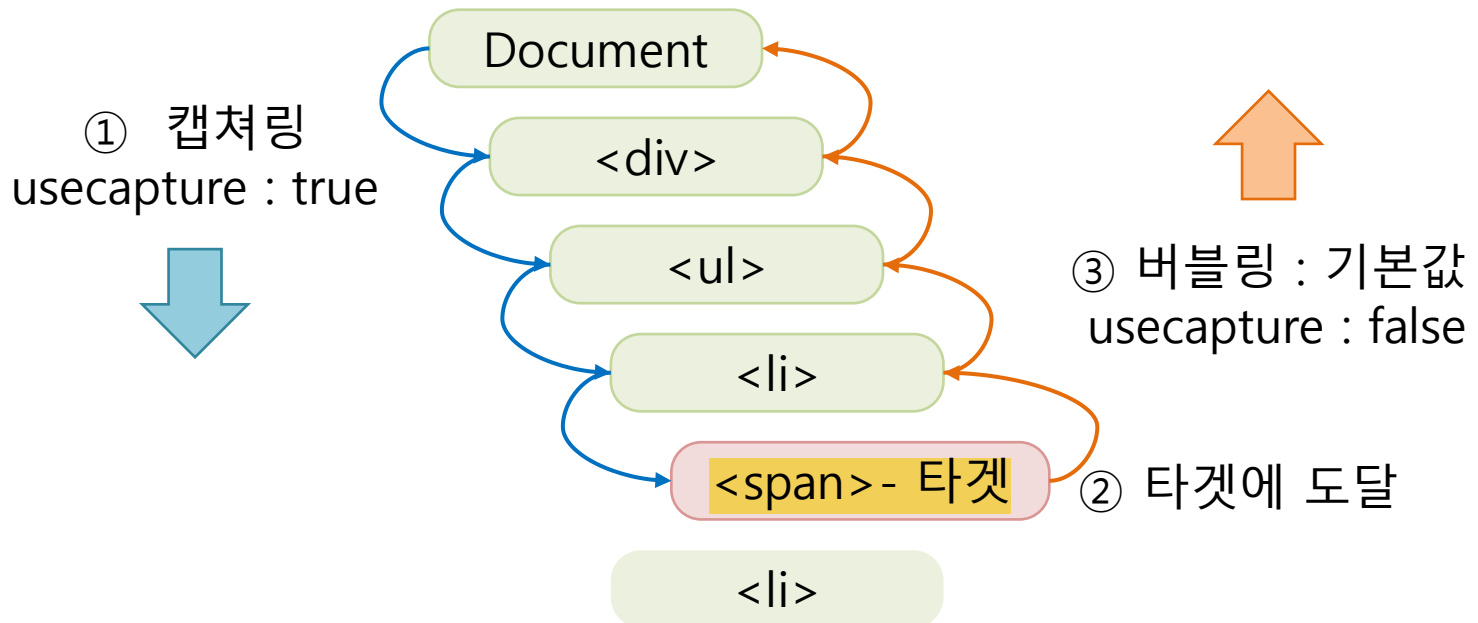
❖ 이벤트(Event)

이벤트 전달

- 이벤트 전달옵션

`addEventListener(eventName, handler, useCapture)`

`eventPropagation(true)` : 캡처링, `(false)` : 버블링



❖ 이벤트(Event)

이벤트 전달을 원하지 않는 경우

- 타겟 요소의 이벤트 리스너의 콜백 함수 마지막 위치에 **event.stopPropagation();** 메서드 호출

```
<h1 id="heading">header
  <p id="paragraph">paragraph</p>
</h1>
document.getElementById('heading').onclick=function(){
  alert('header');
};
document.getElementById('paragraph').onclick=function(){
  alert('paragraph');
  event.stopPropagation();
};
```

❖ document 객체(DOM)

객체 노드 생성

| | 메서드 | 설명 |
|-----|--------------------|-----------------------------------|
| 1 | createElement() | Element node 생성 |
| 2-1 | createTextNode() | Text node 생성 |
| | appendChild() | Text node를 element node 자식 노드로 추가 |
| 2-2 | createAttribute() | 속성 노드 생성 |
| | setAttributeNode() | 속성 노드를 요소 노드에 연결 |
| 3 | appendChild() | 생성된 요소 노드를 부모 노드에 추가 |

❖ document 객체(DOM)

```
//<p class="accent">주문 완료</P>
```

```
// 1. element node : <p>
```

```
let newP = document.createElement('p');
```

```
// 2. text node : 주문 완료
```

```
let newText = document.createTextNode('주문 완료');
```

```
// 3. 자식노드 추가
```

```
newP.appendChild(newText);
```

```
// 4. 부모노드에 추가
```

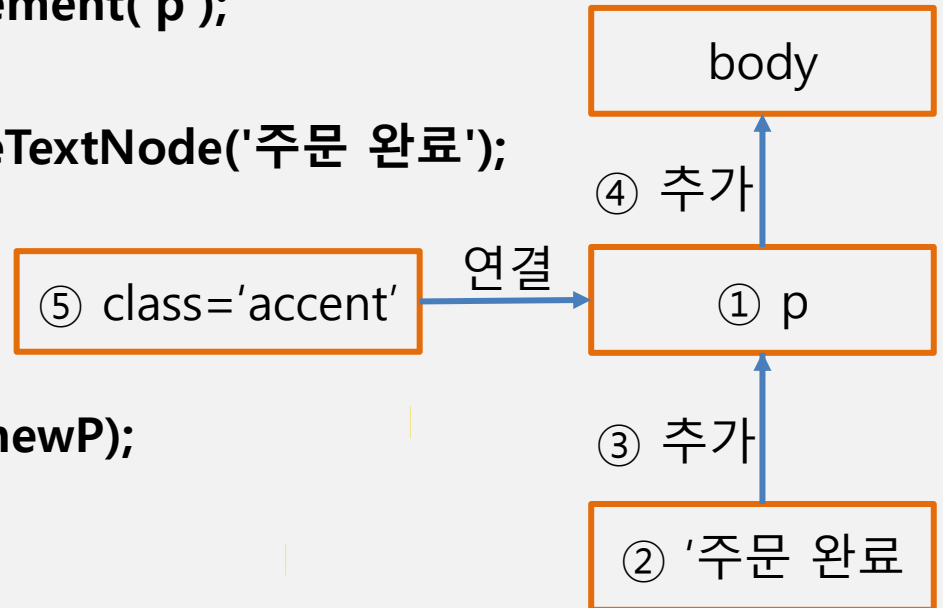
```
document.body.appendChild(newP);
```

```
// 5. 속성 노드 생성, 연결
```

```
let newAttr = document.createAttribute('class');
```

```
newAttr.value = 'accent';
```

```
newP.setAttributeNode(newAttr); newP.setAttribute('class', 'accent');
```



❖ document 객체(DOM)

폼 요소 접근

- Id, class 속성
- name
 - form(name="ship")과 form 요소(name="shipping")에 name 속성이 있어야 함
 - document.ship.shipping.value -> text 상자에 입력된 값
 - document.forms["ship"].elements["shipping"].value
- 폼 배열 (forms)
 - id, class, name 속성이 모두 없을 때 사용
 - 폼 태그를 가져와 배열 형태로 반환
 - document.forms[0].elements[0].value -> 첫 번째 폼의 첫 번째 요소 값

❖ document 객체(DOM)

선택 항목 및 옵션항목 접근 : **select**

- form name = "testForm"
- select name = "major"

`document.testForm.major.options[2].innerText` - 화면표시 내용

`document.testForm.major.options[2].value` -- 서버 전달 값

- 사용자가 선택한 옵션
 - option 은 배열 형태로 저장

```
let selectMenu=document.testForm.major;
```

```
selectMenu.options[selectMenu.selectedIndex].innerText;
```


❖ document 객체(DOM)

라디오 버튼과 체크상자 접근

- name 으로 접근
- form name = "testForm"
- radio name = "subject"
document.testForm.subject
 - radioButtonList(n) 형태로 저장
 - 체크된 값 가져오기

document.querySelector("input[name='subject']:checked").value

- Checkbox name = 'mailing'
document.testForm.mailing
document.querySelectorAll("input[name='mailing']:checked")

❖ 과제 : 프로그램 연습



1.배열.html



2.구구단.html



3.배경색 변경.html



4.계산기.html



5. 좌석배치도.html



6.표만들기.html



7.로또번호.html



8.기념일 계산.html



9.회원가입.html



10.도서목록.html