

19. 문서 객체 모델(DOM)



19-1 문서 객체 모델 알아보기

19-2 DOM 요소에 접근하고 속성 가져오기

19-3 DOM에서 이벤트 처리하기

19-4 DOM에서 노드 추가, 삭제하기

문서 객체 모델 알아보기

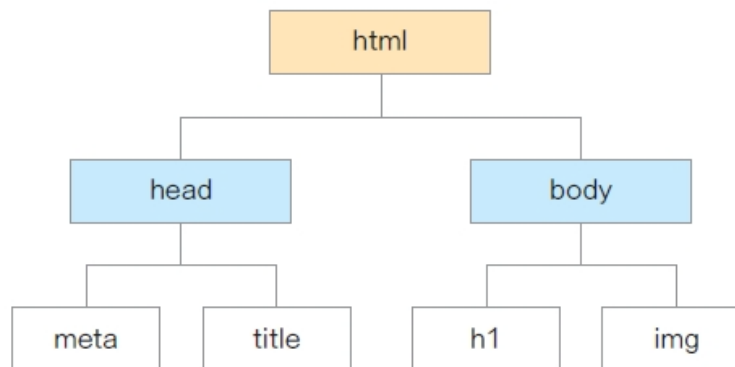
문서 객체 모델이란

자바스크립트를 이용하여 웹 문서에 접근하고 제어할 수 있도록 객체를 사용해 웹 문서를 체계적으로 정리하는 방법

웹 문서와 그 안의 모든 요소를 '객체'로 인식하고 처리함

예) 웹 문서 전체는 document 객체, 삽입한 이미지는 image 객체

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>DOM Tree 알아보기</title>
</head>
<body>
  <h1>Do it!</h1>
  
</body>
</html>
```



문서 객체 모델 알아보기

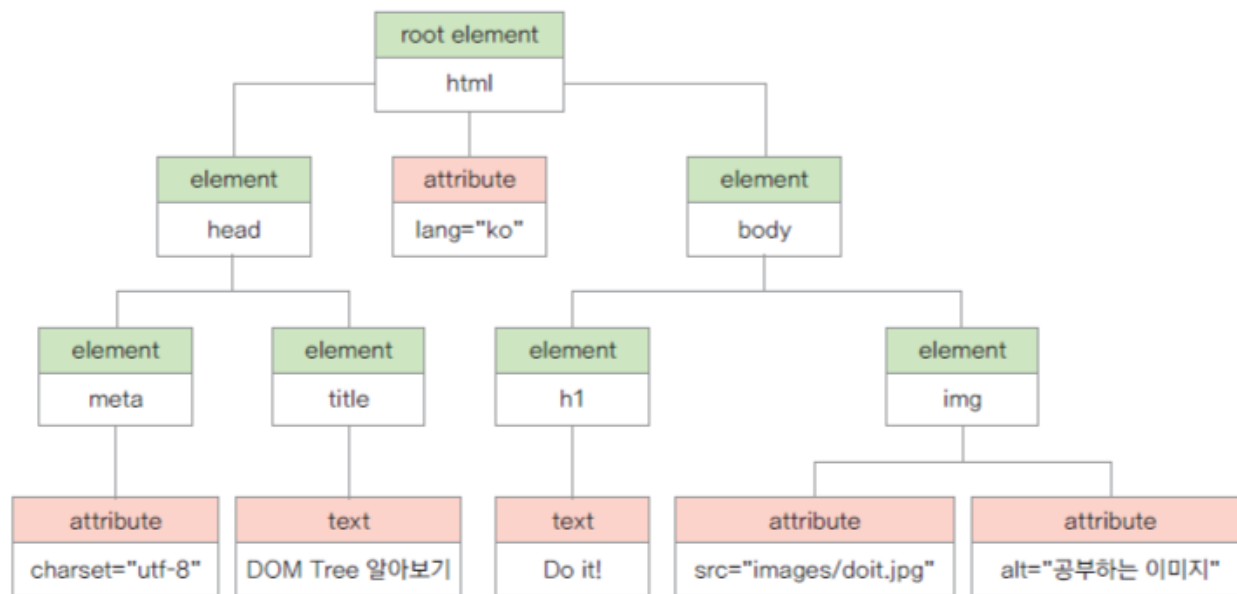
DOM 트리

- 웹 문서에 있는 요소들 간의 부모, 자식 관계를 계층 구조로 표시한 것
- 나무 형태가 되기 때문에 "DOM 트리"라고 함.
- 노드(node) : DOM 트리에서 가지가 갈라져 나간 항목
- 루트 노트(root node) : DOM 트리의 시작 부분(html)

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>DOM Tree 알아보기</title>
</head>
<body>
  <h1>Do it!</h1>
  
</body>
</html>
```

DOM 을 구성하는 원칙

- 모든 HTML 태그는 요소(element) 노드
- 웹 문서의 텍스트 내용은 요소 노드의 자식 노드인 텍스트(text) 노드
- 태그의 속성은 요소 노드의 자식 노드인 속성(attribute) 노드
- 주석은 주석(comment) 노드



DOM 요소에 접근하고 속성 가져오기

DOM 요소에 접근하기

웹 문서에서 원하는 요소를 찾아가는 것을 “접근한다(access)”고 함

getElement-* 함수 사용하기

```
기본형 document.getElementById("id명")
document.getElementsByClassName("class명")
document.getElementsByTagName("태그명")
```

getElementsByClassName()과 getElementsByTagName()은
반환 값이 2개 이상일 수 있음 → HTMLCollection 객체에 저장됨
->

querySelector() 함수

```
기본형 document.querySelector(선택자)
```

- 한 개의 값만 반환
- id 이름 앞에는 해시 기호(#), class 이름 앞에는 마침표(.), 태그는 기호 없이 태그명 사용
- querySelectorAll() 메서드는 반환 값이 여러 개일 때 모두 반환
→ 노드 리스트로 저장됨
->

DOM 요소에 접근하고 속성 가져오기

h1 제목에 접근하기

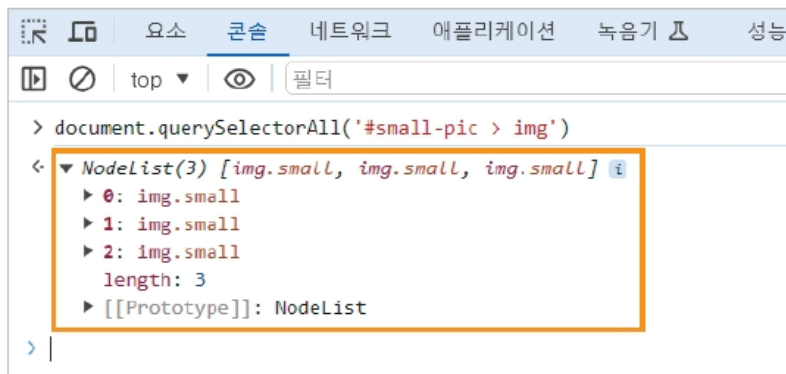


```
document.querySelector('#heading')
```

#small-pic 인 영역의 이미지에 접근하기



```
document.querySelectorAll('#small-pic > img')
```



DOM 요소 내용 가져오고 수정하기

텍스트 내용에 접근하기 및 수정하기

- innerText : 웹 브라우저 창에 보이는 텍스트 내용
- innerHTML : 화면에 보이는 것과 상관없이 HTML 태그까지 포함한 텍스트 내용
- textContent : 화면에 보이는 것과 상관없이 텍스트 내용

기본형 요소.innerText
요소.innerHTML
요소.textContent

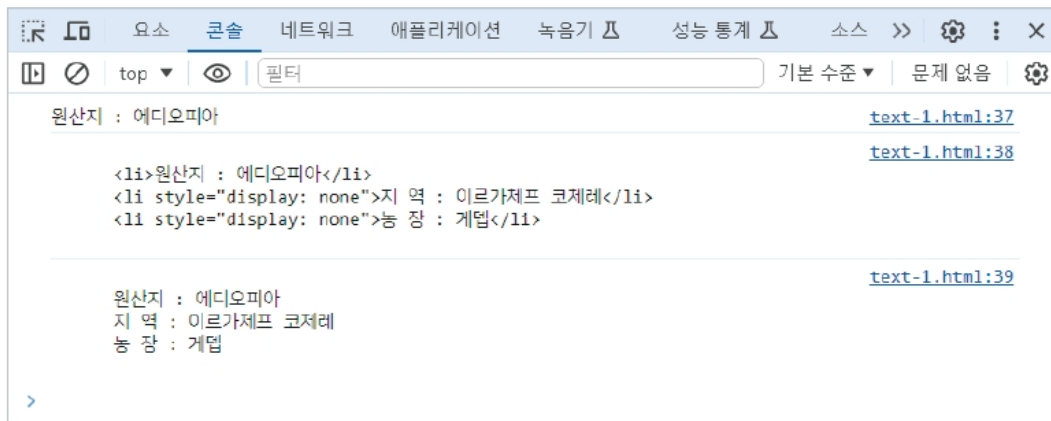
기본형 요소.innerText = 내용
요소.innerHTML = 내용
요소.textContent = 내용

이미지 파일 경로 수정하기

기본형 이미지요소.src = 파일 경로

```
<div id="detail">
  <ul>
    <li>원산지 : 에디오피아</li>
    <li style="display: none">지 역 : 이르가체프 코체레</li>
    <li style="display: none">농 장 : 게덱</li>
  </ul>
</div>

<script>
  let detail = document.querySelector('#detail > ul');
  console.log(detail.innerText);
  console.log(detail.innerHTML);
  console.log(detail.textContent);
</script>
```



DOM 요소 내용 가져오고 수정하기

```
<div class="container">
  <h1 id="heading">에디오피아 게덱</h1>
  
</div>

<script>
  let heading = document.querySelector('#heading');
  let cup = document.querySelector('#cup');
  heading.onclick = () => heading.innerText = '추천! 오늘의 커피';
  cup.onclick = () => cup.src = "images/coffee-blue.jpg";
</script>
```

특정 요소를 클릭했을 때 함수 실행하려면

→ 요소를 가져와 변수에 저장한 후 변수 다음에 .onclick을 붙인 후 함수 연결



DOM에서 이벤트 처리하기

DOM 요소에 함수 직접 연결하기

DOM 요소에 이벤트 처리기 함수를 직접 연결

```
<div class="container">
  
</div>

<script>
  let cat = document.querySelector("#cat");
  cat.onclick = () => alert("이미지를 클릭했습니다");
</script>
```



함수 이름을 사용해 연결하기

- 함수를 따로 정의해 놓았다면 DOM 요소에 함수 이름을 사용해 연결
- 이 때 함수 이름 다음에 괄호를 추가하지 않음

```
<div class="container">
  
</div>

<script>
  let cat = document.querySelector("#cat");
  cat.onclick = changePic; // cat을 클릭하면 changePic 함수 실행

  function changePic() {
    cat.src = "images/kitty-2.png";
  }
</script>
```



DOM에서 이벤트 처리하기

DOM의 event 객체

웹 문서에서 이벤트 발생한 요소가 무엇인지,
어떤 이벤트가 발생했는지 등의 정보가 담긴 객체

```
<div class="container"> </div>
```

```
<script>
```

```
let container = document.querySelector(".container");
container.onclick = (event) => {
  alert(`이벤트 발생 위치: ${event.pageX}, ${event.pageY}`);
}
```

```
</script>
```



	구분	설명
프로퍼티	altKey	이벤트가 발생할 때 (Alt) 를 눌렀는지 여부를 boolean값으로 반환합니다.
	button	마우스에서 누른 버튼의 키값을 반환합니다.
	charCode	keypress 이벤트가 발생할 때 어떤 키를 눌렀는지 유니코드값으로 반환합니다.
	clientX	이벤트가 발생한 가로 위치를 반환합니다.
	clientY	이벤트가 발생한 세로 위치를 반환합니다.
	ctrlKey	이벤트가 발생했을 때 (Ctrl) 를 눌렀는지 여부를 boolean값으로 반환합니다.
	pageX	현재 문서 기준으로 이벤트가 발생한 가로 위치를 반환합니다.
	pageY	현재 문서 기준으로 이벤트가 발생한 세로 위치를 반환합니다.
	screenX	현재 화면 기준으로 이벤트가 발생한 가로 위치를 반환합니다.
	screenY	현재 화면 기준으로 이벤트가 발생한 세로 위치를 반환합니다.
	shiftKey	이벤트가 발생할 때 (Shift) 를 눌렀는지 여부를 boolean값으로 반환합니다.
	target	이벤트가 최초로 발생한 대상을 반환합니다.
	timeStamp	이벤트가 발생한 시간을 반환합니다.
메서드	type	발생한 이벤트 이름을 반환합니다.
	which	키보드와 관련된 이벤트가 발생할 때 키의 유니코드값을 반환합니다.
	preventDefault()	이벤트를 취소할 수 있는 경우에 취소합니다.

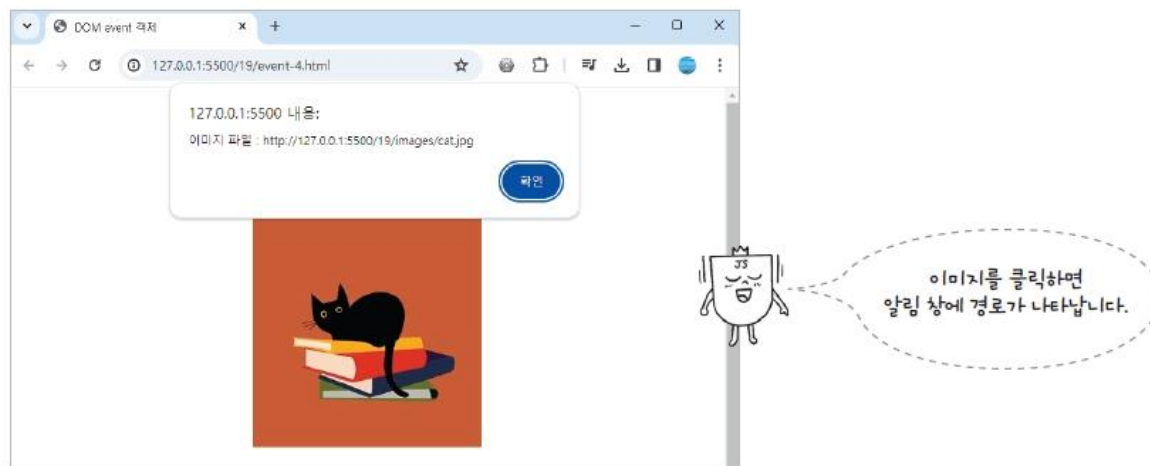
DOM에서 이벤트 처리하기

this 예약어 사용하기

- 이벤트가 발생한 대상에 접근할 때 사용하는 예약어
- 화살표 함수에서는 사용할 수 없음

```
<div class="container">
  
</div>

<script>
  let cat = document.querySelector("#cat");
  cat.onclick = function(event) {
    alert(`이미지 파일: ${this.src}`);
  }
</script>
```



DOM에서 이벤트 처리하기

addEventListener() 함수 사용하기

이벤트 객체를 사용해 이벤트 처리기 연결

기본형 요소.addEventListener(이벤트, 함수, 캡처 여부);

1

2

3

- ❶ 이벤트: 이벤트 유형을 지정합니다. 단, click과 keypress처럼 on을 붙이지 않고 씁니다.
- ❷ 함수: 이벤트가 발생하면 실행할 명령이나 함수를 지정합니다. 여기에서 함수를 정의할 때는 event 객체를 인수로 받습니다.
- ❸ 캡처 여부: 이벤트를 캡처하는지 여부를 지정합니다. true이면 캡처링, false이면 버블링한다는 의미이며 기본값은 false입니다.

DOM에서 이벤트 처리하기

예) 마우스 포인터를 올리면 이미지 바꾸기

```
<div class="container">
  
</div>

<script>
  let cat = document.querySelector("#cat");
  cat.addEventListener("mouseover", changePic, false);
  cat.addEventListener("mouseout", originPic, false);

  function changePic() {
    cat.src = "images/kitty-2.png";
  }
  function originPic() {
    cat.src = "images/kitty-1.png";
  }
</script>
```

함수 표현식으로 사용 가능

```
<div class="container">
  
</div>

<script>
  let cat = document.querySelector("#cat");
  cat.addEventListener("mouseover", () => {
    cat.src = "images/kitty-2.png";
  });
  cat.addEventListener("mouseout", () => {
    cat.src = "images/kitty-1.png";
  });
</script>
```

DOM에서 이벤트 처리하기

CSS 속성에 접근하기

자바스크립트를 사용해 각 요소의 스타일을 자유롭게 수정할 수 있음

기본형 `document.요소명.style.속성명`

예) id가 desc인 요소의 글자를 파란색으로 변경하려면

```
document.querySelector("#desc").style.color = "blue";
```

- color처럼 한 단어인 속성명은 그대로 사용
- background-color, border-radius처럼 중간에 하이픈(-)이 있는 속성은 backgroundColor나 borderRadius처럼 두 단어를 합쳐서 사용
- 이때 두 번째 단어의 첫 글자는 Color와 Radius처럼 대문자로 표시

```
<div id="container">
  <p>도형 위로 마우스 포인터를 올려놓으세요.</p>
  <div id="rect"></div>
</div>

<script>
let rect = document.querySelector("#rect");
rect.addEventListener("mouseover", () => {
  rect.style.backgroundColor = "green"; // rect 요소의 배경색
  rect.style.borderRadius = "50%"; // rect 요소의 테두리 둥글게 처리
});
rect.addEventListener("mouseout", () => {
  rect.style.backgroundColor = ""; // rect 요소의 배경색 지우기
  rect.style.borderRadius = ""; // rect 요소의 테두리 둥글게 처리하지 않음
});
</script>
```



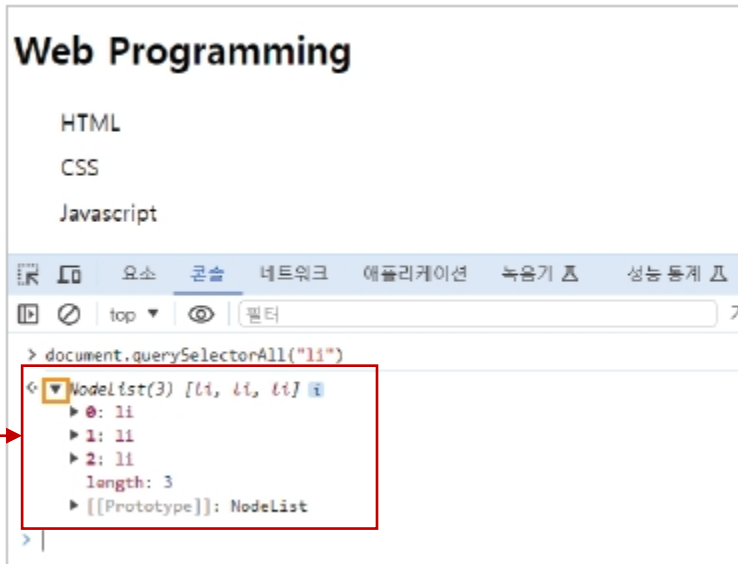
DOM에서 노드 추가, 삭제하기

노드 리스트란

querySelectorAll() 메서드를 사용해 가져온 여러 개의 노드를 저장한 것

```
<h1>Web Programming</h1>  
<ul id="itemList">  
  <li>HTML</li>  
  <li>CSS</li>  
  <li>Javascript</li>  
</ul>
```

```
> document.querySelectorAll("li")
```

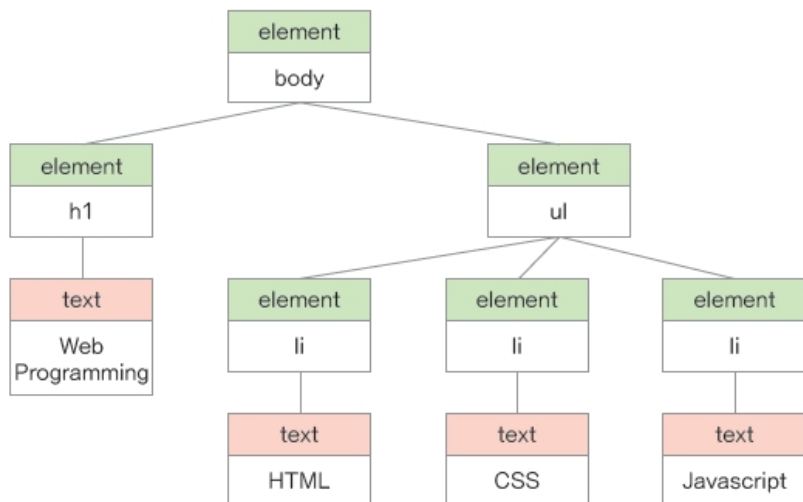


내용이 있는 텍스트 노드 추가하기

기존 문서

- ul 안에 li 3개 있는 구조
- 여기에 새로운 항목 li 추가하기

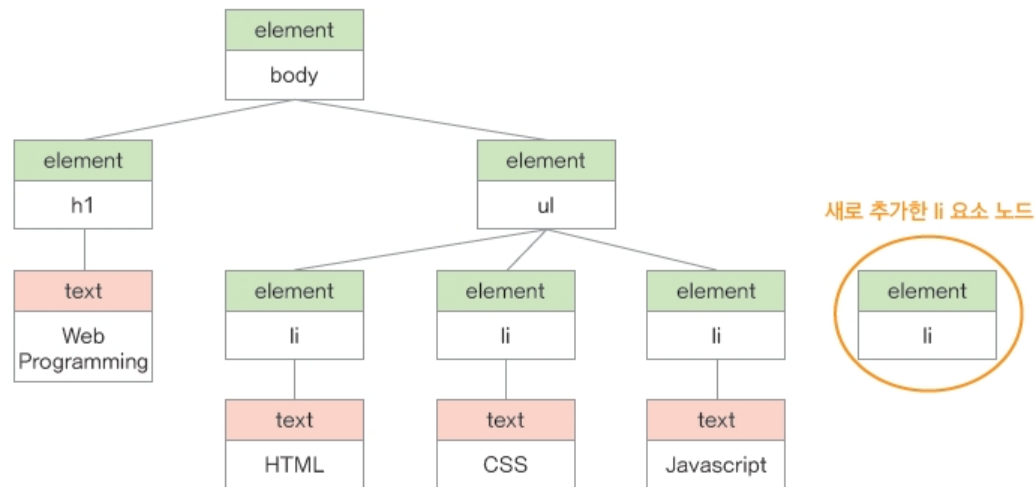
```
<h1>Web Programming</h1>
<ul id="itemList">
  <li>HTML</li>
  <li>CSS</li>
  <li>Javascript</li>
</ul>
```



1. 요소 노드 만들기 - createElement() 메서드

기본형 document.createElement(요소명)

```
let newItem = document.createElement("li")
```

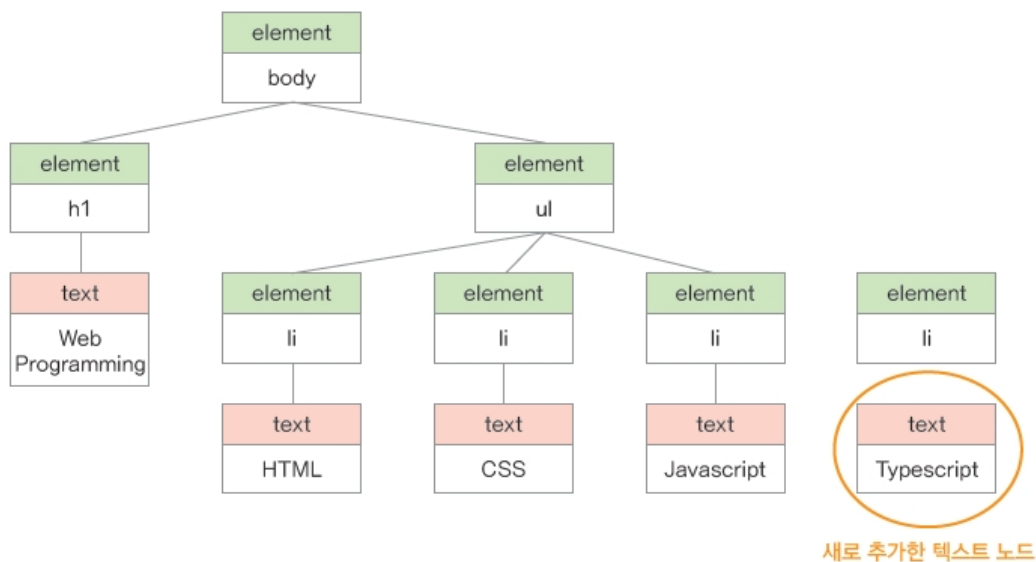


내용이 있는 텍스트 노드 추가하기

2. 텍스트 노드 만들기 – `createTextNode()` 메서드

기본형 `document.createTextNode(내용)`

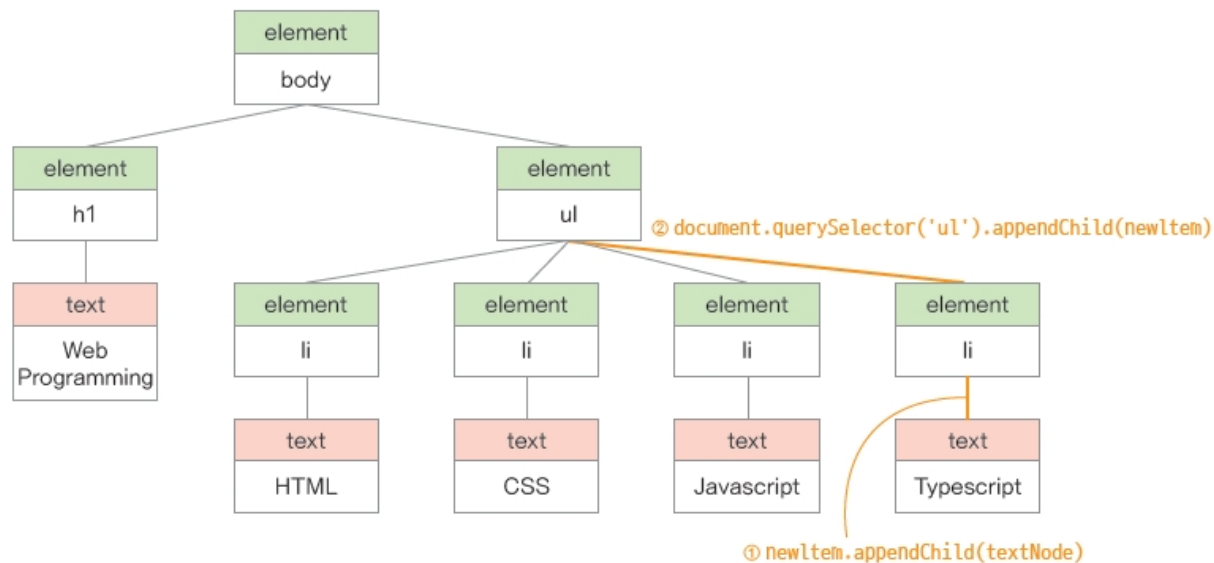
```
let textNode = document.createTextNode("Typescript")
```



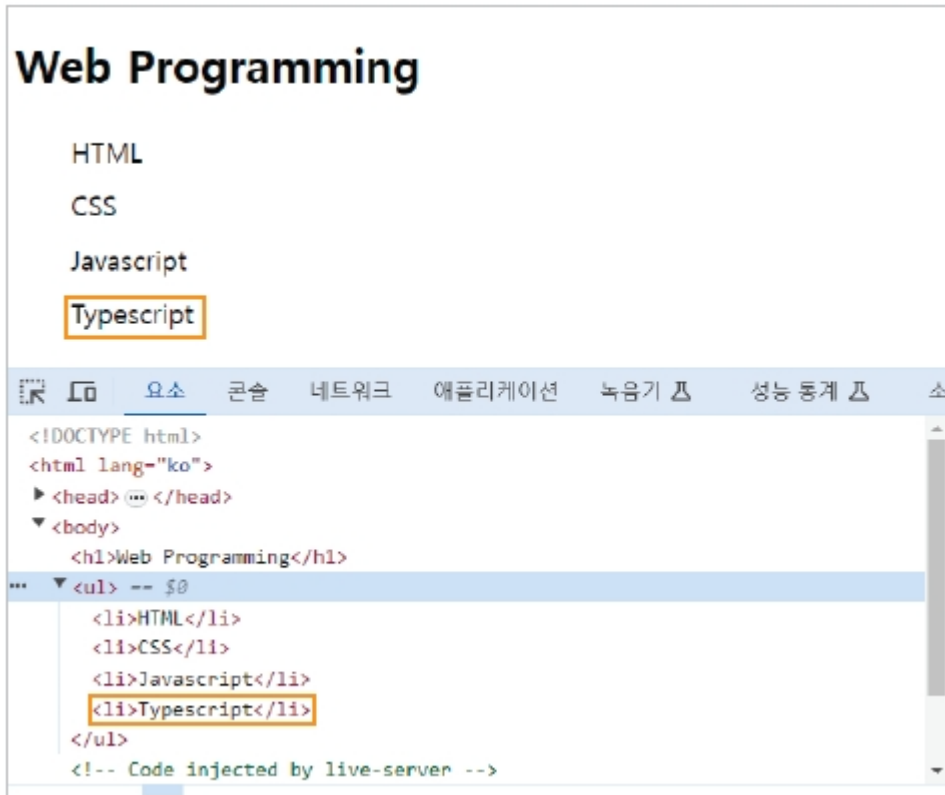
3. 자식 노드 연결하기 – `appendChild()` 메서드

기본형 `부모노드.appendChild(자식노드)`

```
newItem.appendChild(textNode)  
document.querySelector('ul').appendChild(newItem)
```

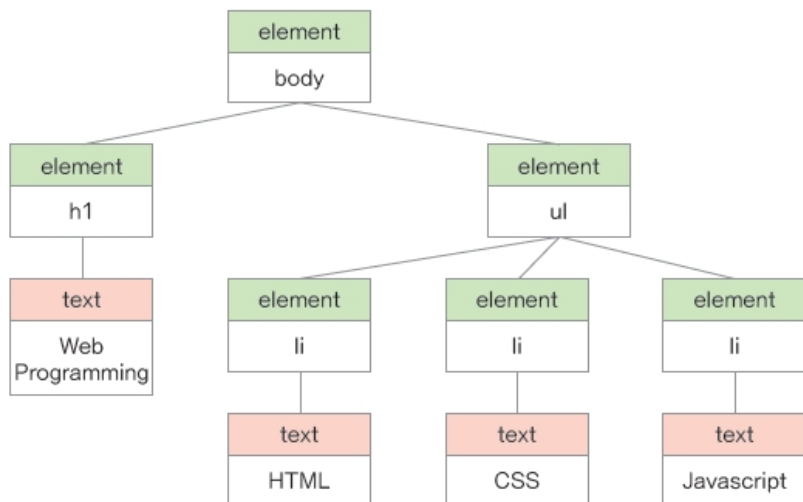


내용이 있는 텍스트 노드 추가하기



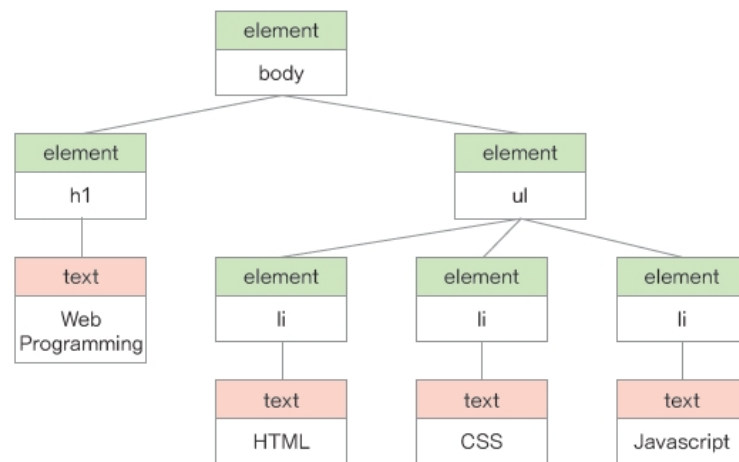
속성 값이 있는 새로운 요소 추가하기

```
<h1>Web Programming</h1>  
<ul id="itemList">  
  <li>HTML</li>  
  <li>CSS</li>  
  <li>Javascript</li>  
</ul>
```



1. 요소 노드 만들기 – createElement() 메서드

```
let newImg = document.createElement("img")
```

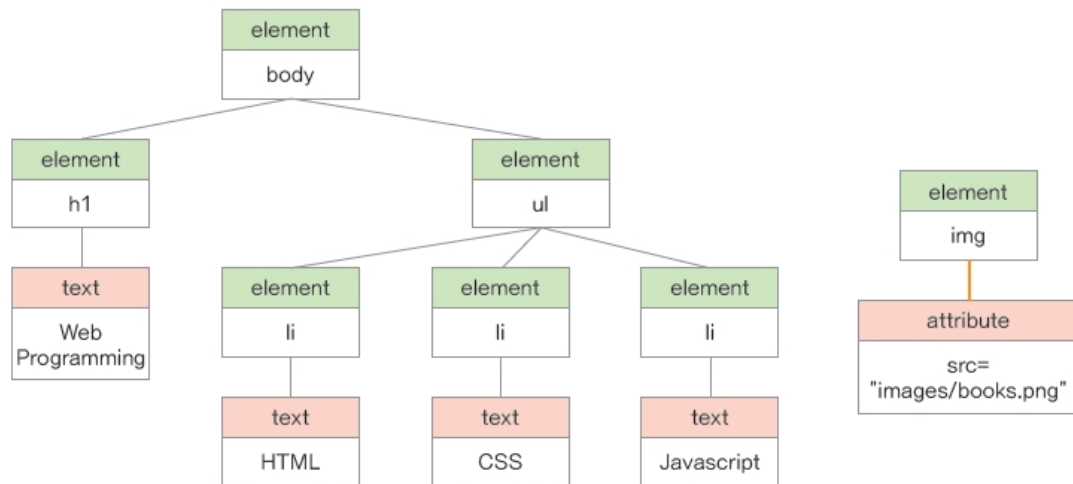


새로 추가한 img 요소 노드

속성 값이 있는 새로운 요소 추가하기

2. 속성 추가하기

```
newImg.src = "images/books.png"
```

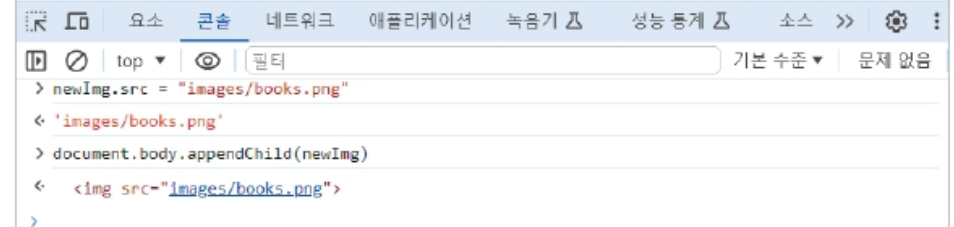


3. 자식 노드 연결하기

```
document.body.appendChild(newImg)
```

Web Programming

HTML
CSS
Javascript

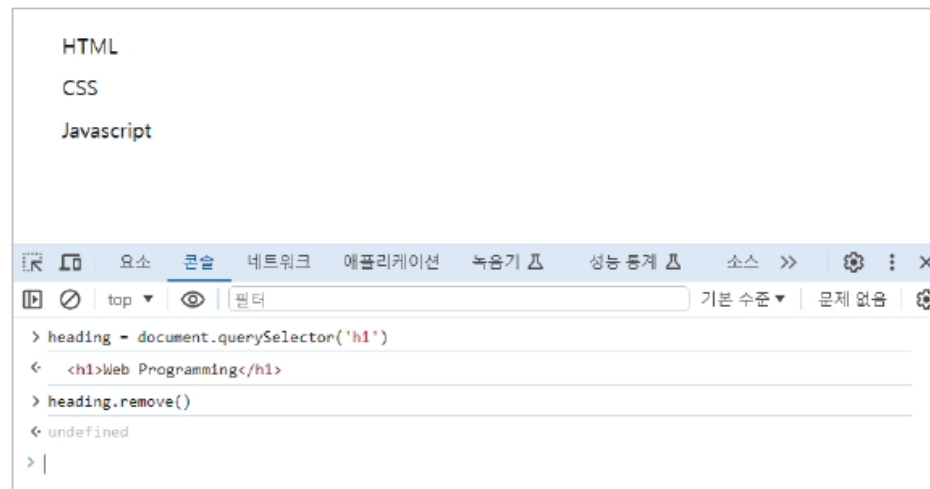


노드 삭제하기

기본형 노드.remove()

```
<h1>Web Programming</h1>
<ul id="itemList">
  <li>HTML</li>
  <li>CSS</li>
  <li>Javascript</li>
</ul>
```

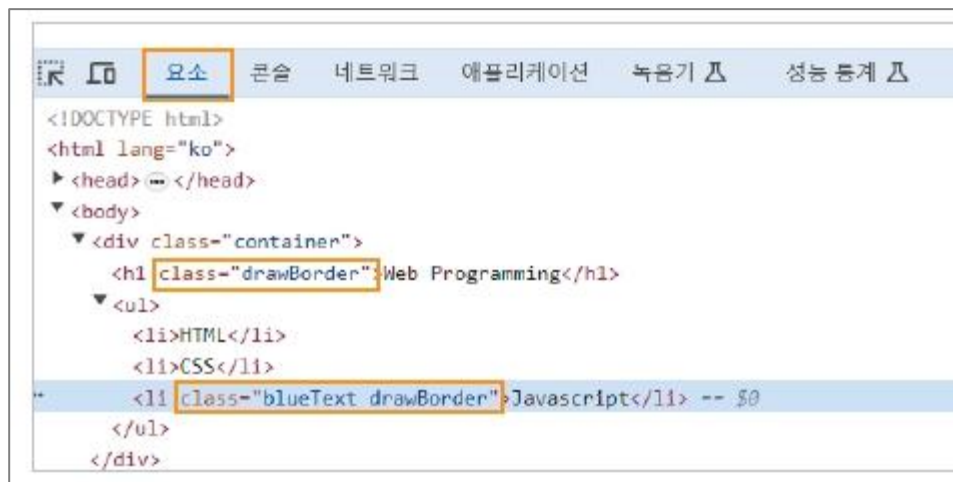
```
heading = document.querySelector('h1')
heading.remove()
```



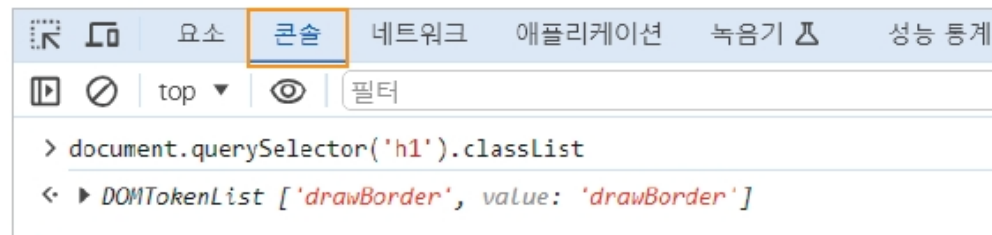
class 속성 추가, 삭제하기

classList 프로퍼티

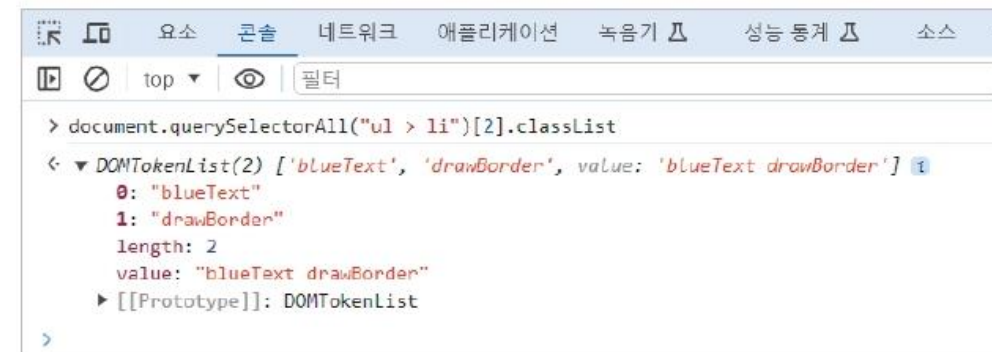
19\classList.html



```
document.querySelector('h1').classList
```



```
document.querySelectorAll("ul > li")[2].classList
```



class 속성 추가, 삭제하기

classList에서 사용하는 함수

함수	설명
add(클래스명)	지정한 클래스를 classList에 추가합니다.
remove(클래스명)	지정한 클래스를 classList에서 제거합니다.
toggle(클래스명)	지정한 클래스가 있으면 classList에서 제거하고, 지정한 클래스가 없으면 classList에 추가합니다.
contains(클래스명)	지정한 클래스가 classList에 있는지 확인합니다.

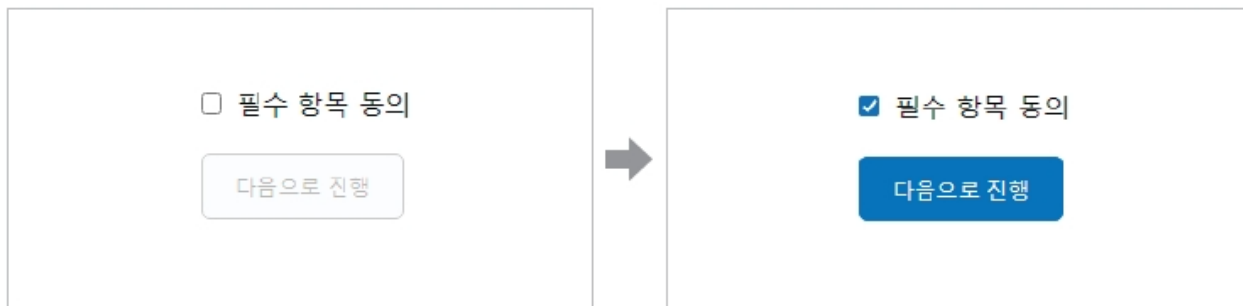
기본형

요소.classList.add(클래스명)

요소.classList.remove(클래스명)

class 속성 추가, 삭제하기

예) 체크 박스 클릭하면 버튼 활성화하기 – if ...else 문 사용



1. 클래스 스타일 정의하기


```
<style>
.disabled {
  color: #ccc;
  background-color: #f0f0f0;
  border: 1px solid #ccc;
}
.enabled {
  color: #fff;
  background-color: #007bff;
  border: 1px solid #007bff;
}
</style>
```

class 속성 추가, 삭제하기

예) 체크 박스 클릭하면 버튼 활성화하기

웹 문서 모두 로드되면 발생하는 이벤트

2. 함수 작성하기



```
<script>
document.addEventListener('DOMContentLoaded', function() {
  const checkbox = document.querySelector('#agree');
  const proceedButton = document.querySelector('#proceed');
  checkbox.addEventListener('change', function() {
    if (this.checked) {
      proceedButton.classList.remove('disabled');
      proceedButton.classList.add('enabled');
      proceedButton.disabled = false;
    } else {
      proceedButton.classList.remove('enabled');
      proceedButton.classList.add('disabled');
      proceedButton.disabled = true;
    }
  });
});
</script>
```


class 속성 추가, 삭제하기

예) 체크 박스 클릭하면 버튼 활성화하기 – toggle() 메서드 사용

```
<script>
document.addEventListener('DOMContentLoaded', function() {
  const checkbox = document.querySelector('#agree');
  const proceedButton = document.querySelector('#proceed');
  checkbox.addEventListener('change', function() {
    proceedButton.classList.toggle('enabled', this.checked); // .enabled 토글
    proceedButton.classList.toggle('disabled', !this.checked); // .disabled 토글
    proceedButton.disabled = !this.checked; // 버튼의 활성화/비활성화 상태
  });
});
</script>
```