LIKE LION

5강

클라이언트 사이드 자바스크립트

WHATEVER YOU WANT, MAKE IT REAL.

강사 정길용

LIKE LION

DOM

Event

{ BOM }

Ajax 프로그래밍

Web APIs

LIKE LION

▶ 웹 브라우저에서 실행되는 자바스크립트 환경

- ECMAScript: 자바스크립트 언어에 대한 표준
  - https://ecma-international.org/publications-and-standards/standards/ecma-262

- DOM(Document Object Model): 웹페이지 제어를 위한 표준
  - https://dom.spec.whatwg.org
  - window.document 등
  - Event

- BOM(Browser Object Model): 웹페이지 외부의 브라우저 기능 제어를 위한 표준
  - HTML 표준: https://html.spec.whatwg.org
  - window.navigator: 브라우저와 운영체제에 대한 정보 제공
  - window.location: 현재 페이지의 URL에 대한 제어(읽기, 수정)
  - window.history: 브라우저의 과거 페이지 이동 정보에 대한 제어(읽기, 수정)
  - alert, setTimeout 등

- Web APIs: 브라우저가 제공하는 웹 기능을 위한 표준
  - https://spec.whatwg.org
  - XMLHttpRequest: 서버와 통신에 사용되는 객체(Ajax)
  - Web Storage, Notifications API, WebSocket 등

▶ DOM(Document Object Model)

- 1998년 10월 W3C에서 DOM 레벨 1 발표
- 2000년 11월 DOM 레벨 2 발표
- 2004년 04월 DOM 레벨 3 발표
- 2015년 11월 DOM 레벨 4 발표
- 2019년 05월 W3C에서 WHATWG으로 이관
    - https://dom.spec.whatwg.org

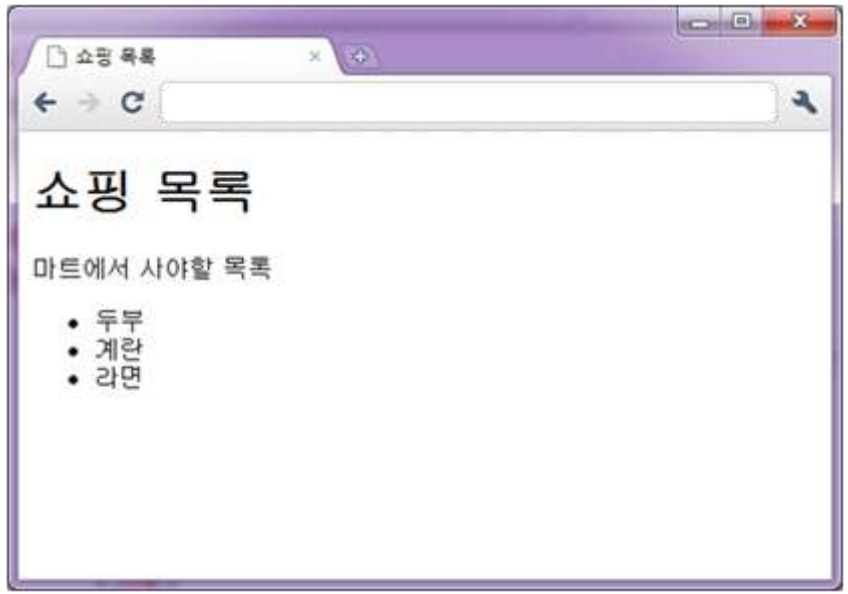
- HTML, XML 등의 문서를 제어하기 위한 방법을 정의

- 텍스트 기반의 HTML, XML 문서를 일정한 규칙에 의해 객체로 만들고 이를 이용하여 문서를 제어(특정 요소를 추출, 삽입, 삭제, 이동 등)

LIKE LION

ex05-01.html

```
<html>
<head>
<title>쇼핑목록</title>
</head>
<body>
  <h1>쇼핑 목록</h1>
  <p>마트에서 사야할 목록</p>
  <ul id="purchases" class="list">
    <li>두부</li>
    <li>계란</li>
    <li>라면</li>
  </ul>
</body>
</html>
```

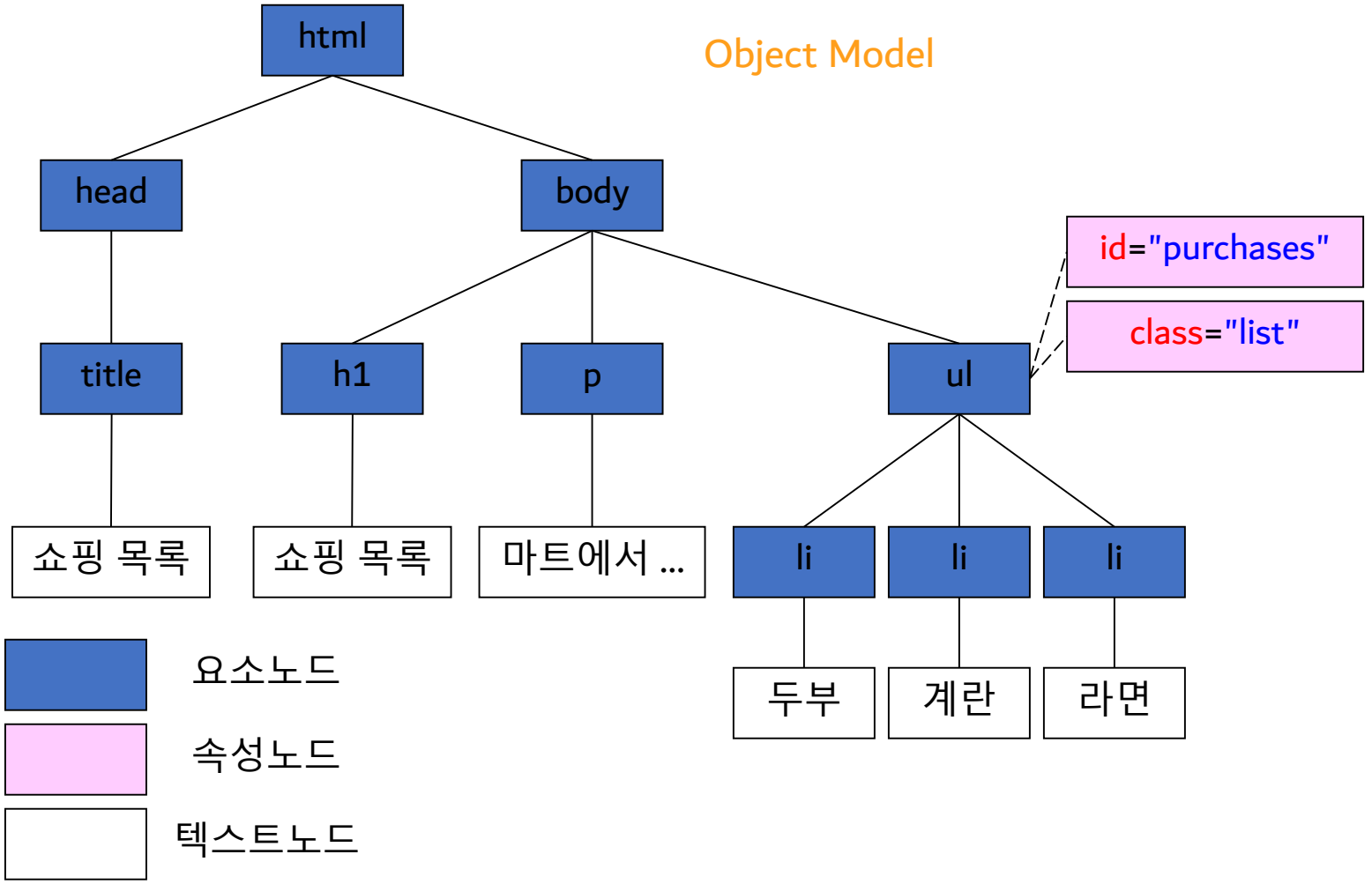


LIKE LION

Document

```
<html>
<head>
<title>쇼핑목록</title>
</head>
<body>
  <h1>쇼핑 목록</h1>
  <p>마트에서 사야할 목록</p>
  <ul id="purchases" class="list">
    <li>두부</li>
    <li>계란</li>
    <li>라면</li>
  </ul>
</body>
</html>
```

Object Model



▶ 노드(Node)

- DOM 트리구조는 모든 구성원이 각각의 객체로 인식되며 이러한 객체 하나 하나를 노드라고 함

▶ 노드의 종류(주로 사용되는 노드)

- 문서노드(document node)
- 요소노드(element node)
- 속성노드(attribute node)
- 텍스트노드(text node)
- ……

종류	설명	nodeName	nodeType	nodeValue
문서 노드	문서	#document	9	null
요소 노드	태그	태그의 이름	1	null
속성 노드	요소의 속성	속성의 이름	2	속성의 값
텍스트 노드	요소의 내용	#text	3	문자열 값

LIKE LION

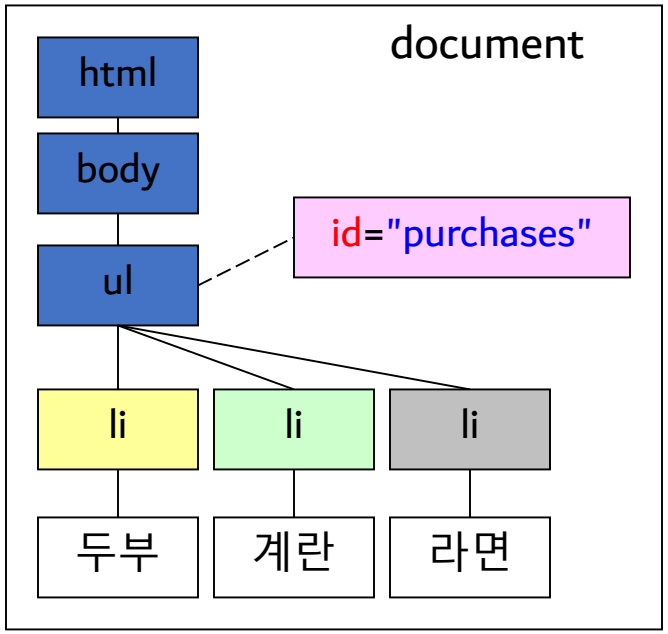
ex05-01.html

▶ 태그의 id를 이용하여 노드 찾기

- document.getElementById(id)
  - id 속성값에 해당하는 노드객체를 반환

```
const purchases = document.getElementById("purchases");
```

ul



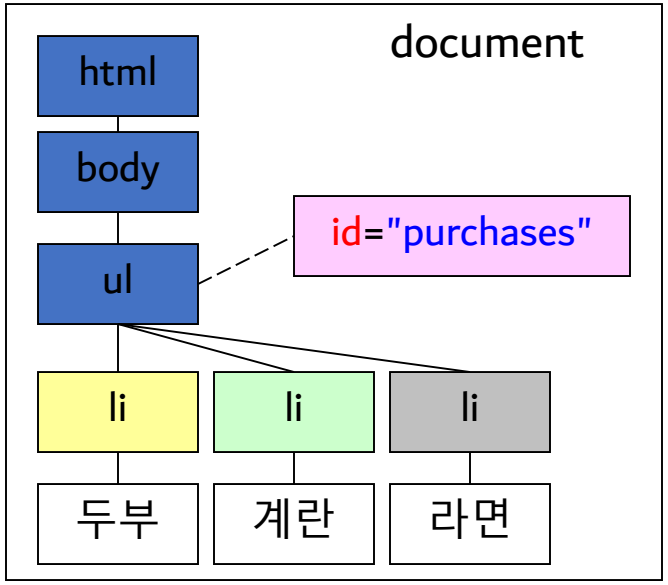
LIKE LION

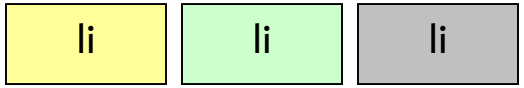
ex05-01.html

▶ 태그명을 이용하여 노드 찾기

- 요소노드.getElementsByTagName(tagName)
  - 지정한 요소노드의 하위 모든 요소를 대상으로 태그명(tagName)에 해당하는 요소노드를 배열로 반환(tagName에 "*"을 지정하면 모든 요소를 배열로 반환)

```
const liList = purchases.getElementsByTagName("li");
```





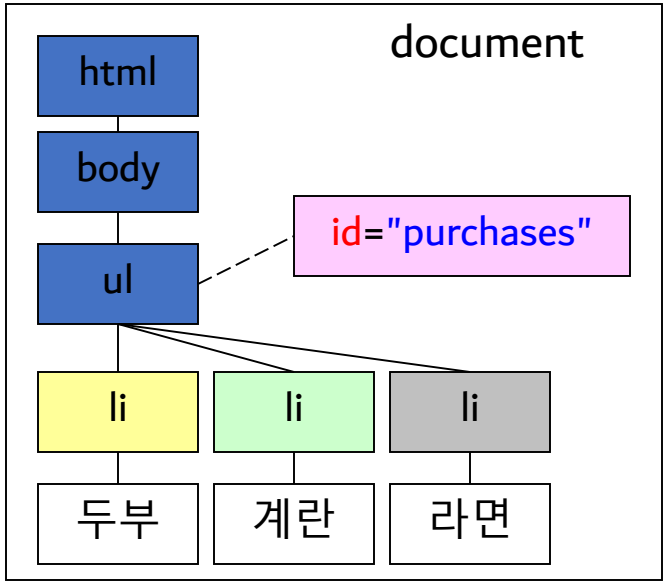
LIKE LION

ex05-01.html

▶ 트리구조를 이용하여 노드 찾기

- 부모/자식 노드 찾기

요소노드의 속성	설명
childNodes	자식 노드가 배열 형태로 저장
firstChild	첫번째 자식 노드(요소, 텍스트, 주석)
firstElementChild	첫번째 자식 요소 노드
lastChild	마지막 자식 노드(요소, 텍스트, 주석)
lastElementChild	마지막 자식 요소 노드
parentNode	부모 노드




```
const purchases = document.getElementById('purchases');
const firstItem = purchases.firstElementChild;
const lastItem = purchases.lastElementChild;
const liList = purchases.childNodes;
```

ul

li

li

li li li

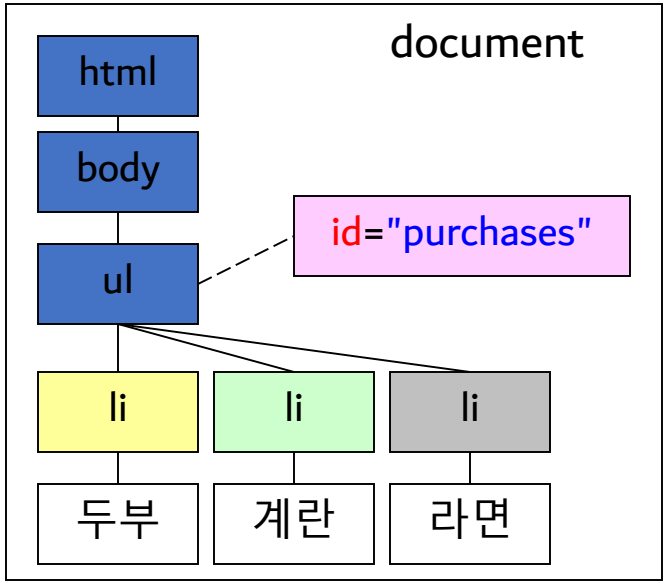
LIKE LION

ex05-01.html

▶ 트리구조를 이용하여 노드 찾기

- 형제 노드 찾기

속성	설명
previousSibling	바로 앞의 형제 노드(요소, 텍스트, 주석)
previousElementSibling	바로 앞의 형제 요소 노드
nextSibling	바로 뒤의 형제 노드(요소, 텍스트, 주석)
nextElementSibling	바로 뒤의 형제 요소 노드




```
const purchases = document.getElementById("purchases");
const secondItem = purchases.childNodes[3];
const firstItem = secondItem.previousElementSibling;
const lastItem = secondItem.nextElementSibling;
```

ul

li

li

li

LIKE LION

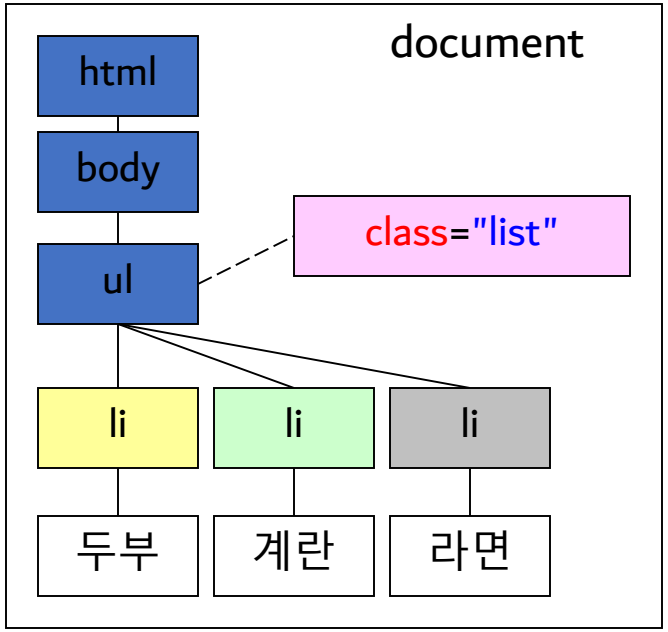
ex05-01.html

▶ class 속성으로 노드 찾기

- document.getElementsByClassName(className)
  - class 속성값이 className인 요소 노드의 목록을 반환

```
const purchases = document.getElementsByClassName("list")[0];
```

ul



LIKE LION

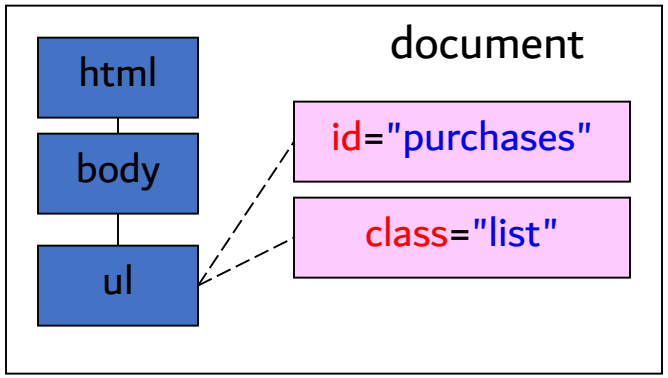
ex05-01.html

▶ CSS 셀렉터 이용

- Selector: CSS에서 사용하는 노드 선택 구문
  - https://www.w3.org/TR/css3-selectors/
- document.querySelector(selector)
  - 지정한 selector 구문에 매칭되는 노드 목록 중 첫번째 노드를 반환
- document.querySelectorAll(selector)
  - 지정한 selector 구문에 매칭되는 노드 목록을 반환

```
var purchases = document.querySelector(".list");
var purchases = document.querySelector("#purchases");
var purchases = document.querySelectorAll("ul")[0];
```

ul



LIKE LION

ex05-01.html 

elem.innerHTML

- elem의 내부 HTML 코드의 값을 조회하거나 수정
- elem 자신은 제외

```
const shoppingList =
document.querySelector('#purchases');
console.log(shoppingList.innerHTML);
```

```
'₩n    <li>두부</li>₩n    <li>계란</li>₩n    <li>
라면</li>₩n  '
```

elem.outerHTML

- elem의 내부 HTML 코드의 값을 조회하거나 수정
- elem 자신을 포함

```
const shoppingList =
document.querySelector('#purchases');
console.log(shoppingList.outerHTML);
```

```
'<ul id="purchases" class="list">₩n    <li>두부
</li>₩n    <li>계란</li>₩n    <li>라면</li>₩n
</ul>'
```

```
<ul id="purchases" class="list">
  <li>두부</li>
  <li>계란</li>
  <li>라면</li>
</ul>
```

LIKE LION

ex05-01.html 

▶ elem.textContent

- elem의 내부 텍스트 노드의 값을 조회하거나 수정
- 소스코드의 값 그대로 조회

```
const secondLi =
document.querySelector('#purchases > li:nth-
child(2)');
console.log(secondLi.textContent); // 계란□
✓□
```

```
<ul id="purchases" class="list">
  <li>두부<span>✓□</span></li>
  <li>계란<span hidden>✓□</span></li>
  <li>라면<span>✓□</span></li>
</ul>
```

▶ elem.innerText

- elem의 내부 텍스트 노드의 값을 조회하거나 수정
- 브라우저에 의해서 실제 보이는 값으로 조회
- 화면에 보이지 않는 요소는 제외(hidden)

```
const secondLi =
document.querySelector('#purchases > li:nth-
child(2)');
console.log(secondLi.innerText); // 계란
```

- 두부 ✔
- 계란
- 라면 ✔

LIKE LION

ex05-01.html 

▶ 노드 생성

- document 객체의 createXxx() 메소드를 이용

메소드	설명
createElement(nodeName)	지정한 태그명으로 요소노드 생성
createTextNode(nodeValue)	지정한 내용으로 텍스트노드 생성
createAttribute(attributeName)	지정한 이름으로 속성노드 생성

```
const newLiNode = document.createElement("li");

const newTextNode = document.createTextNode("우유");
```

li

우유

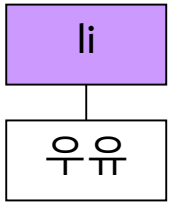
LIKE LION

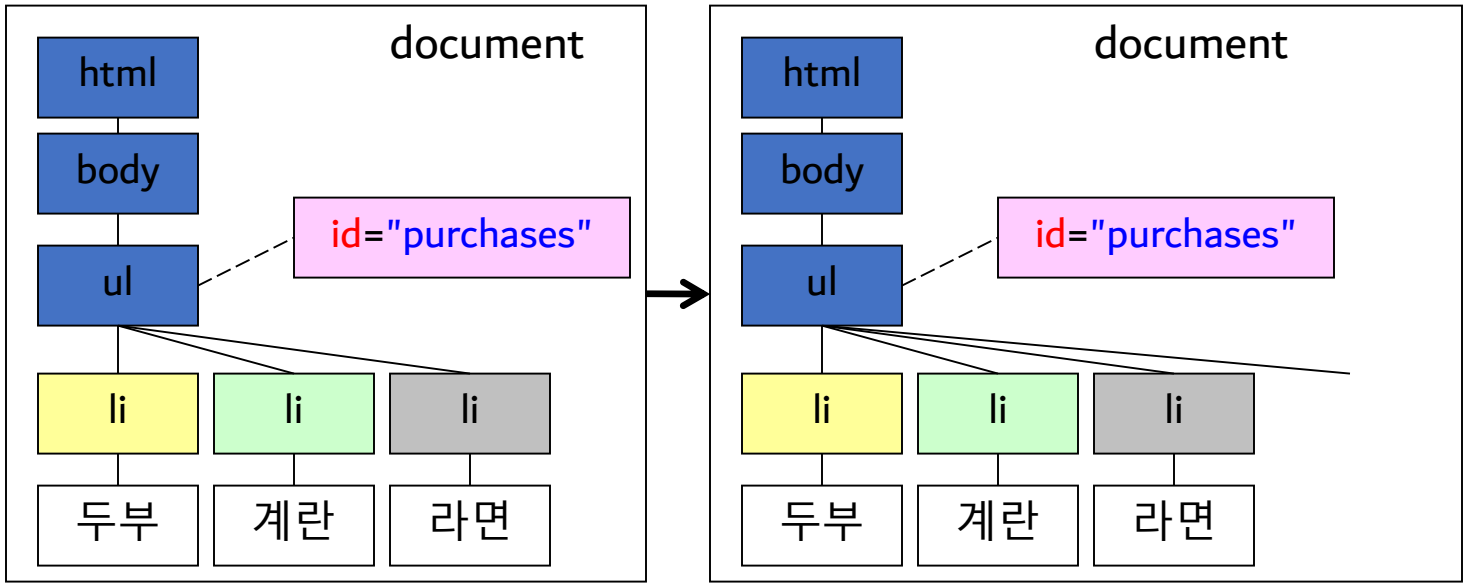
ex05-01.html

▶ 노드 추가

- 요소노드.appendChild(childNode)
  - 지정한 노드를(childNode) 요소노드의 마지막 자식노드로 추가

```
☞ const newLiNode = document.createElement("li");
☞ const newTextNode = document.createTextNode("우유");
☞ newLiNode.appendChild(newTextNode);
☞ purchases.appendChild(newLiNode);
```





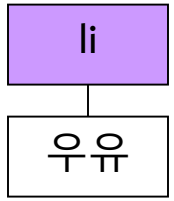
LIKE LION

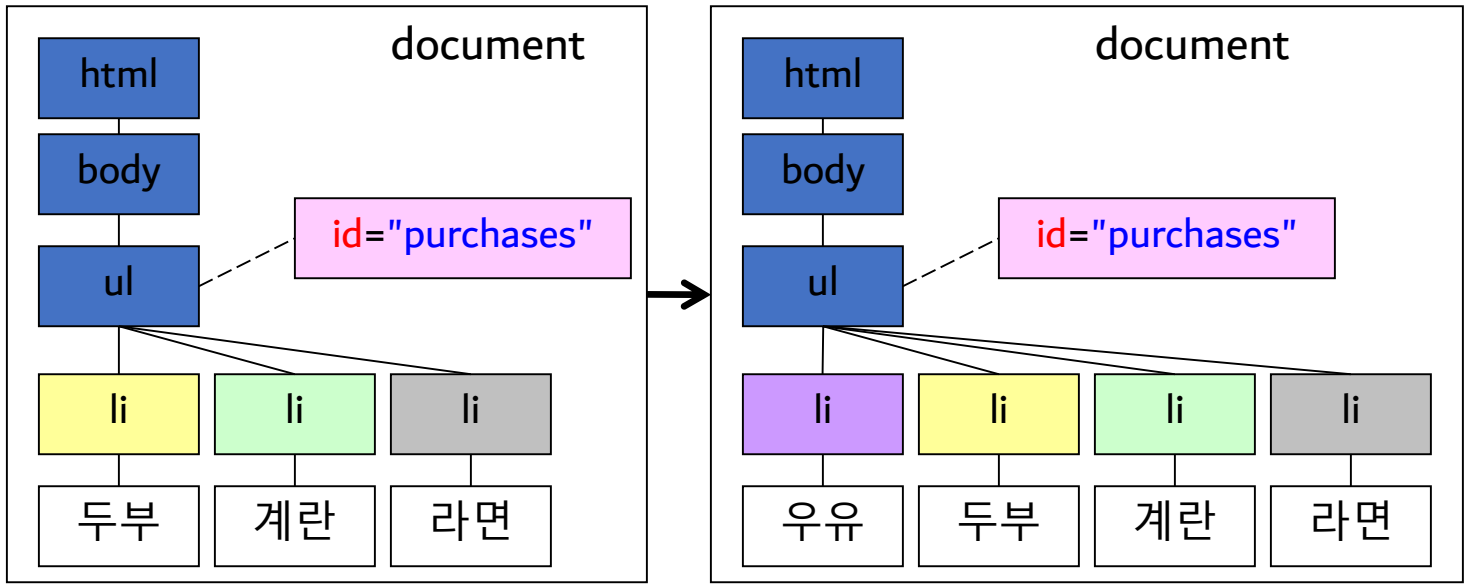
ex05-01.html

▶ 노드 삽입

- 요소노드.insertBefore(newNode, targetNode)
  - 지정한 노드를(newNode) targetNode 앞에 삽입

```
☞ const purchases = document.getElementById("purchases");
☞ purchases.insertBefore(newLiNode, purchases.firstChild);
```





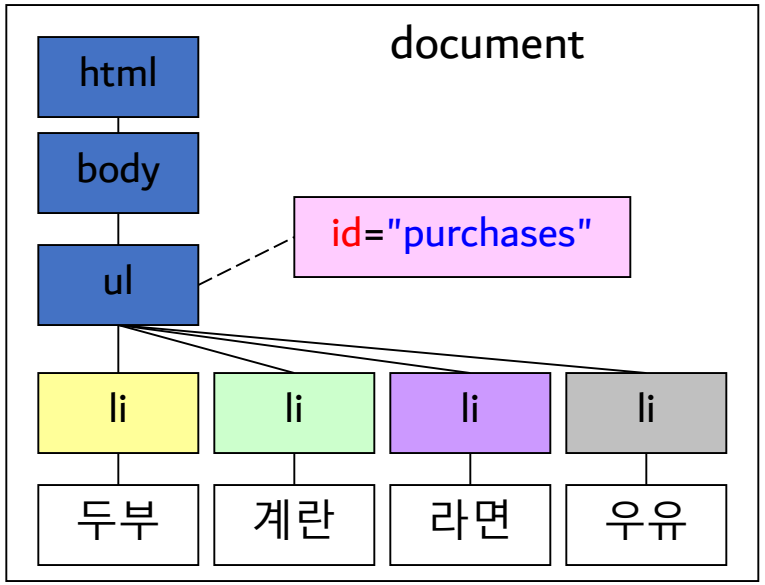
LIKE LION

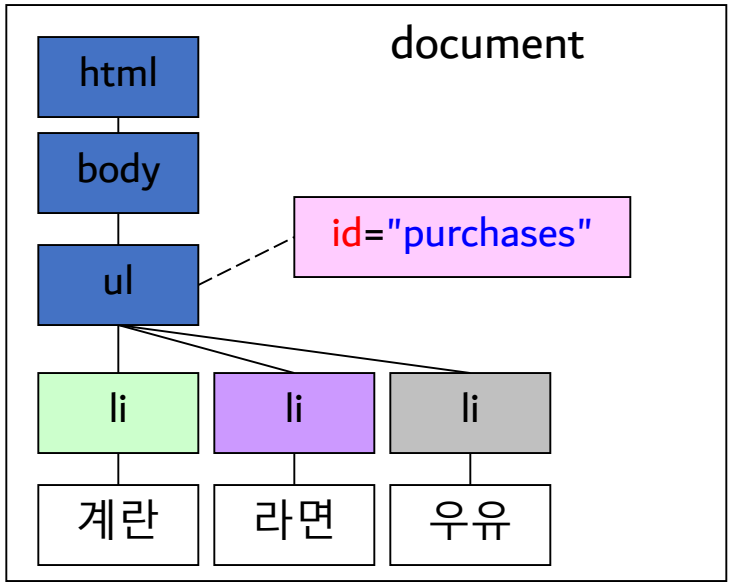
ex05-01.html

▶ 노드 삭제

- 요소노드.removeChild(childNode)
  - 지정한 노드를(childNode) 삭제한다.

```
☞ const purchases = document.getElementById("purchases");
☞ purchases.removeChild(purchases.firstChild);
```





LIKE LION

ex05-01.html

▶ 노드 복사

- 노드.cloneNode(haveChild)
  - 지정한 노드를 복사한다. haveChild가 true이면 하위 모든 노드를 같이 복사하고 false이면 지정한 노드만 복사한다.

```
const purchases = document.getElementById("purchases");
const cloneLi = purchases.firstChild.cloneNode(true);
purchases.appendChild(cloneLi);
```