KB

It's Your Life

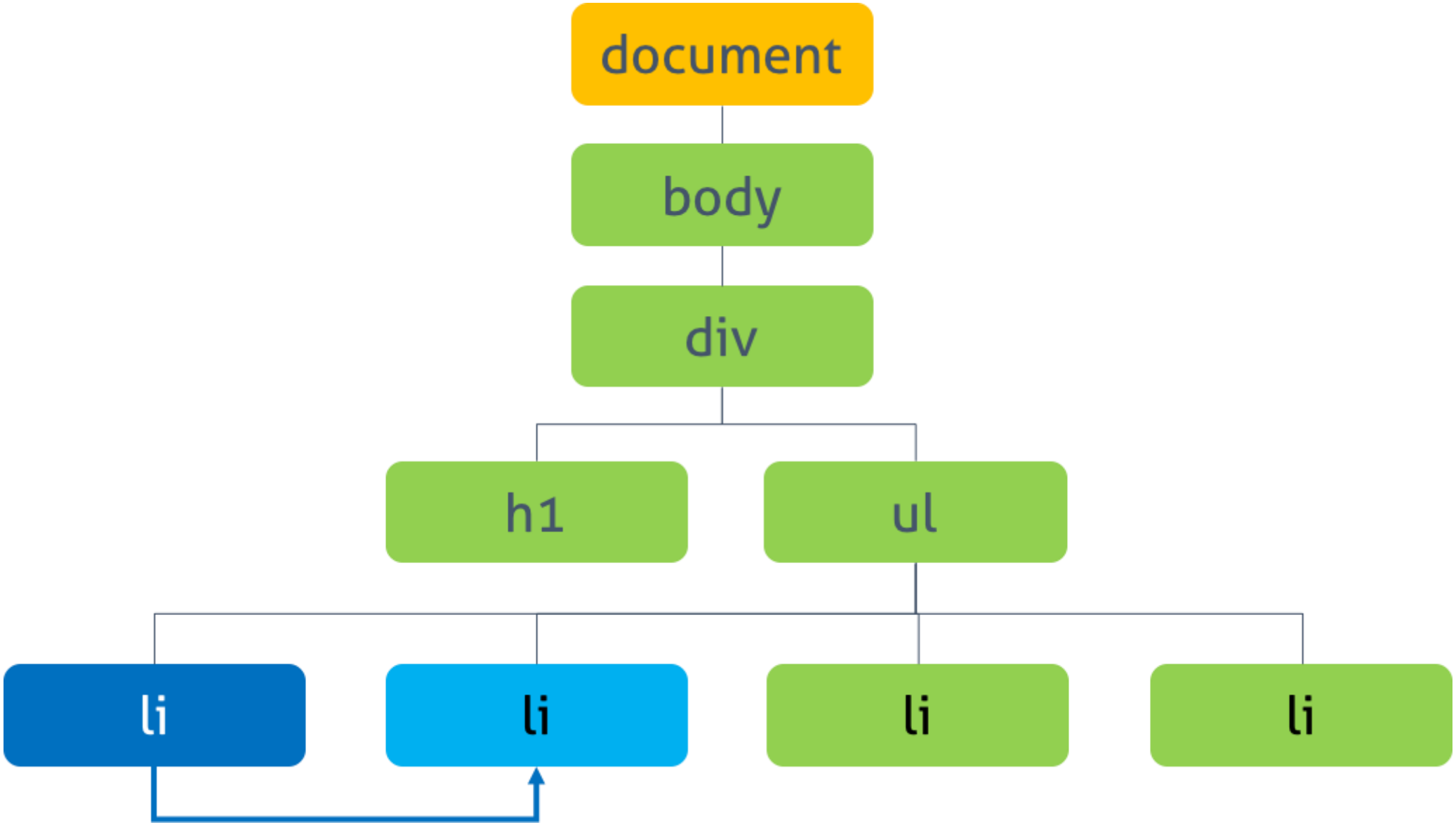
with

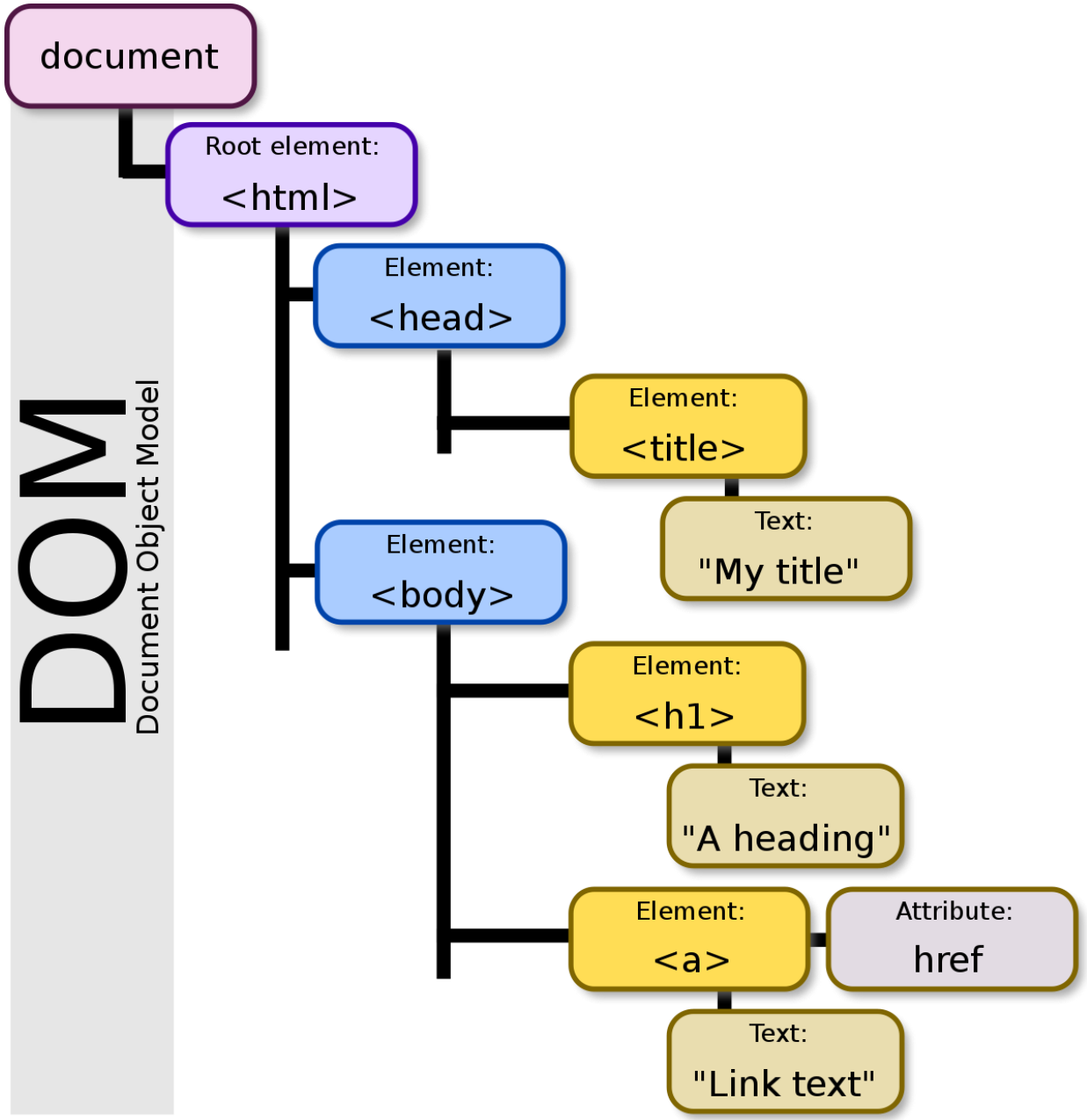
DOM

(Document Object Model)

DOM(Document Object Model)

- HTML 문서 요소의 집합!
- HTML 문서는 각각의 node 와 object 의 집합으로 문서를 표현
- 따라서 각각 node 또는 object 에 접근하여 문서 구조 / 스타일 / 내용 등을 변경 할 수 있도록 하는 것!


document
body
div
h1
ul
li
li
li
li


DOM
Document Object Model
document
Root element:
<html>
Element:
<head>
Element:
<title>
Text:
"My title"
Element:
<body>
Element:
<h1>
Text:
"A heading"
Element:
<a>
Attribute:
href
Text:
"Link text"

DOM API

Document Object Model, Application Programming Interface

```
// HTML 요소(Element) 1개 검색/찾기
const boxEl = document.querySelector('.box');

// HTML 요소에 적용할 수 있는 메소드!
boxEl.addEventListener();

// 인수(Arguments)를 추가 가능!
boxEl.addEventListener(1, 2);

// 1 - 이벤트(Event, 상황)
boxEl.addEventListener('click', 2);

// 2 - 핸들러(Handler, 실행할 함수)
boxEl.addEventListener('click', function () {
  console.log('Click~!');
});
```

querySelector

querySelector(“요소 선택자”)

- 요소 선택자를 사용해서 자신이 가져오고 싶어하는 요소를 가져오는 메소드
- 문서에서 만나는 제일 첫번째 요소를 반환 합니다!

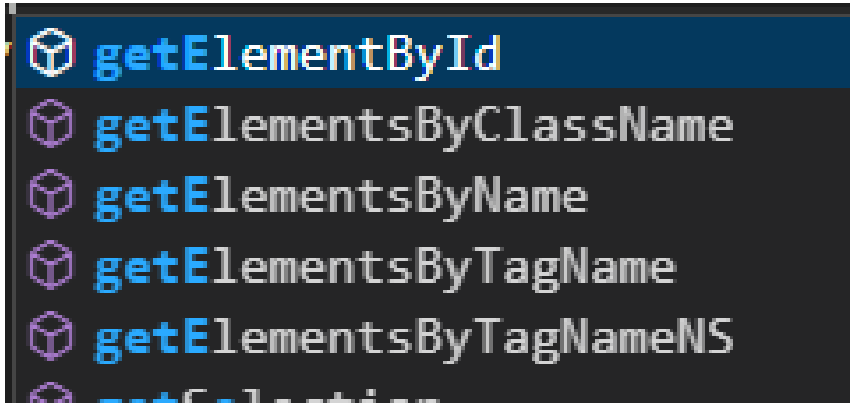
```
let boxEl = document.querySelector(".box");

console.log(boxEl);
```

getElementById

getElementById(“ID”)

- ID 이름을 불러서 해당 ID를 가지는 요소를 불러오는 메소드

```
const inputEl = document.getElementById("input");
```



querySelectorAll

querySelectorAll("요소 선택자")

- 문서에 존재하는 모든 요소를 찾아주는 메소드
- 모든 요소를 가져와서 배열(같은) 데이터로 만들어 줍니다!

```
<body>
    <div class="box">1</div>
    <div class="box">2</div>
    <div class="box">3</div>
    <div class="box">4</div>
    <div class="box">5</div>
    <div class="box">6</div>
    <div class="box">7</div>
</body>
```

```
let boxEls = document.querySelectorAll(".box");
console.log(boxEls);
```

```
▼ NodeList(7) [div.box, div.box, div.box, div.box, div.box, div.box, div.box]
  ▶ 0: div.box
  ▶ 1: div.box
  ▶ 2: div.box
  ▶ 3: div.box
  ▶ 4: div.box
  ▶ 5: div.box
  ▶ 6: div.box
    length: 7
  ▶ [[Prototype]]: NodeList
```

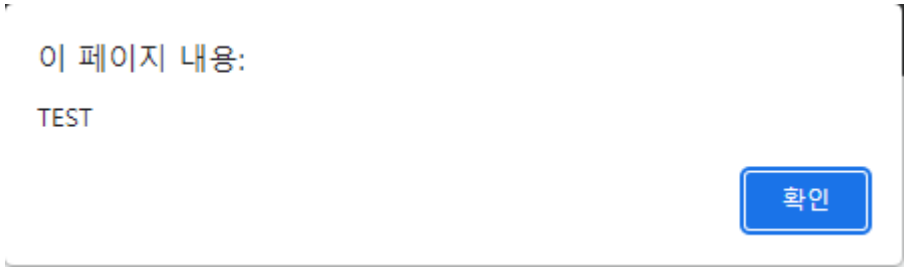
Onclick!

onclick

- 각각의 HTML 요소에 속성 값으로 JS 함수를 연결

```html
<body>
    <div class="box" onclick="test();">click</div>
</body>
```

```js
function test() {
    alert("TEST");
}
```



AddEventListener

addEventListener(이벤트, 명령)

- 선택 요소에 지정한 이벤트가 발생하면, 약속 된 명령어를 실행시키는 메소드

```
let boxEl = document.querySelector(".box");

console.log(boxEl);

boxEl.addEventListener("click", function() {
    alert("click!");
})
```

```
document.querySelector(".box").addEventListener("click", function() {
    alert("click");
})
```

```html
<head>
  <style>
    .box {
      width: 100px;
      height: 100px;
      background-color: skyblue;
    }
  </style>
</head>
<body>
  <div class="box">1</div>
  <script>
    const boxEl = document.querySelector('.box');

    boxEl.addEventListener('click', function () {
      alert('click!');
    });
  </script>
</body>
```

addEventListener 이벤트의 종류

- Click : 클릭
- Mouse 계열
  - Mouseover : 요소에 커서를 올렸을 때
  - Mouseout : 마우스가 요소를 벗어날 때
  - Mousedown : 마우스 버튼을 누르고 있는 상태
  - Mouseup : 마우스 버튼을 떼는 순간
- Focus : 포커스가 갔을 때
- Blur : 포커스가 벗어나는 순간

addEventListener 이벤트의 종류

- Key 계열
  - Keypress : 키를 누르는 순간 + 누르고 있는 동안 계속 발생
  - Keydown : 키를 누르는 순간에만 발생
  - Keyup : 키를 눌렀다가 떼는 순간
- Load : 웹페이지에 필요한 모든 파일(html, css, js 등)의 다운로드가 완료 되었을 때
- Resize : 브라우저 창의 크기가 변경 될 때
- Scroll : 스크롤이 발생할 때
- Unload : 링크를 타고 이동하거나, 브라우저를 닫을 때
- Change : 폼 필드의 상태가 변경 되었을 때

this 와

e.target

this 활용하기

- DOM 요소에서 this 를 사용하면, this 는 자기 자신 Node 를 가르킵니다!

  → onclick 에서 주로 사용!

```
<ul>
  <li onclick="showThis(this);">1</li>
  <li onclick="showThis(this);">2</li>
  <li onclick="showThis(this);">3</li>
  <li onclick="showThis(this);">4</li>
</ul>
```

```
function showThis(t) {
  console.log(t);
}
```

```
▶ <li onclick="showThis(this);">…</li>
```

e.target 활용하기

- Document 에서 발생하는 이벤트는 이벤트 객체 e 를 통해 확인 할 수 있습니다. → addEventListener 에서 주로 사용!

```html
<ul>
  <li class="list">1</li>
  <li>2</li>
  <li>3</li>
  <li>4</li>
</ul>
```

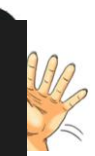
```js
const list = document.querySelector(".list");

list.addEventListener("click", function (e) {
  console.log(e.target);
});
```

```
▶ <li class="list">…</li>
```

classList

classList.add / remove / contain

- 선택 요소에 class 를 더하거나, 빼거나, 클래스가 존재하는지 체크하는 메소드
- 해당 기능과 CSS 를 잘 활용하면 변화무쌍(?)한 웹페이지 구성이 가능



```html
<head>
  <style>
    .box {
      width: 100px;
      height: 100px;
      border: 1px solid black;
    }
    .orange {
      background-color: orange;
    }
  </style>
</head>
<body>
  <div class="box"></div>
  <script>
    const boxEl = document.querySelector('.box');
    boxEl.addEventListener('click', function () {
      if (boxEl.classList.contains('orange')) {
        boxEl.classList.remove('orange');
      } else {
        boxEl.classList.add('orange');
      }
    });
  </script>
</body>
```

setAttribute

setAttribute, html 요소 속성 추가

- 선택한 요소의 속성 값을 직접 지정할 수 있는 메소드
- 요소.setAttribute(“속성명”, “지정할 속성“)

```
searchInputEl.setAttribute("placeholder", "통합검색");
```

```
boxEl.setAttribute("style", "background-color: orange");
```

textContent

```html
<body>
    <div class="box"></div>
    <script>
      let boxEl = document.querySelector('.box');
      console.log(boxEl.textContent);

      boxEl.textContent = `KB It's your Life`;
      console.log(boxEl.textContent);
    </script>
  </body>
```

```
                                                    lecture.html:12
KB It's your Life                                   lecture.html:14
```

createElement

createElement, html 요소 생성

- Html DOM 요소를 만들어내는 메소드
- document.createElement(“요소 이름”)

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

```
const box = document.createElement("div");
```

append,

appendChild

append / appendChild, 요소 붙이기

- 특정 DOM 요소에 다른 요소를 자식으로 붙이는 메소드
- DOM요소.append(추가할 내용)
- DOM요소.appendChild(추가할 요소)

```
const li = document.createElement("li");
const checkBtn = document.createElement("input");
checkBtn.setAttribute("type", "checkbox");
li.append(checkBtn);
```

append()

- append() 는 Node 와 String 을 전부 추가 할 수 있어요
- append() 는 여러 가지 값을 한번에 붙일 수 있어요
- append() 는 반환(리턴) 값이 없어요!

appendChild()

- appendChild() 는 Node 만 추가할 수 있어요
- appendChild() 는 한 번에 하나만 추가할 수 있어요
- appendChild() 는 추가한 Node 를 반환(리턴) 합니다!

remove

remove, 선택한 DOM 을 지우는 메소드

- DOM요소.remove

```
<ul>
     <li>1</li>
     <li class="list">2</li>
     <li>3</li>
     <li>4</li>
   </ul>
```

```
const list = document.querySelector(".list");
console.log(list.);
```

- 1
- 3
- 4

parentNode

parentNode, 부모 요소 확인하기

- 특정 DOM 요소의 부모 노드를 가져오는 메소드
- DOM요소.parentNode

```
<ul>
    <li>1</li>
    <li class="list">2</li>
    <li>3</li>
    <li>4</li>
  </ul>
```

```
const list = document.querySelector(".list");
console.log(list.parentNode);
```

```
▼<ul>
  ▶<li>…</li>
  ▶<li class="list">…</li>
  ▶<li>…</li>
  ▶<li>…</li>
  </ul>
```

childNodes

childNodes, 자식 요소 확인하기

- 특정 DOM 요소의 자식 요소를 전부 확인하는 메소드
- DOM요소.childNodes

```html
<ul>
    <li>1</li>
    <li class="list">2</li>
    <li>3</li>
    <li>4</li>
  </ul>
```

```
const list = document.querySelector("ul");
console.log(list.childNodes);
```

```
tt.html:57
NodeList(9) [text, li, text, li.list, text, li, text, li, text]
  0: text
  1: li
  2: text
  3: li.list
  4: text
  5: li
  6: text
  7: li
  8: text
  length: 9
  [[Prototype]]: NodeList
```

children

children, 자식 요소 확인하기

- 특정 DOM 요소의 자식 노드만을 확인하는 메소드
- DOM요소.children

```
<ul>
    <li>1</li>
    <li class="list">2</li>
    <li>3</li>
    <li>4</li>
  </ul>
```

```
const list = document.querySelector("ul");
console.log(list.childNodes);
```

```
▼HTMLCollection(4) [li, li.list, li, li] i
  ▶0: li
  ▶1: li.list
  ▶2: li
  ▶3: li
    length: 4
  ▶[[Prototype]]: HTMLCollection
```

Defer,

Async


그런데
말입니다
SBS

dom.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Document</title>
    <script src="./dom.js"></script>
</head>

<body>
    <div class="box">Box!!</div>
</body>

</html>
```

```
null
```

dom.js

```js
let boxEl = document.querySelector(".box");

console.log(boxEl);
```

dom.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Document</title>
</head>

<body>
    <div class="box">Box!!</div>
    <script src="./dom.js"></script>
</body>

</html>
```

```html
<div class="box">Box!!</div>
```

dom.js

```js
let boxEl = document.querySelector(".box");

console.log(boxEl);
```

dom.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Document</title>
    <script defer src="./dom.js"></script>
</head>

<body>
    <div class="box">Box!!</div>
</body>

</html>
```

```html
<div class="box">Box!!</div>
```

dom.js

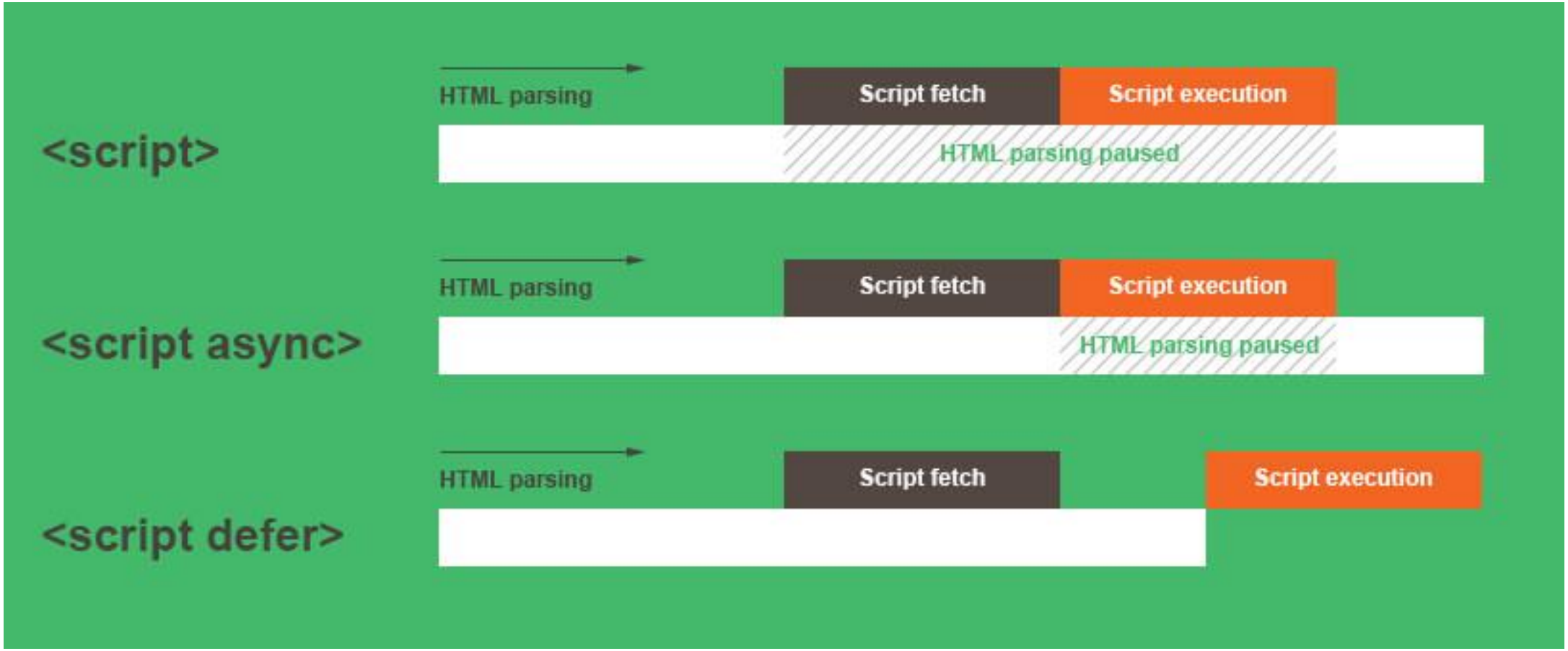
```js
let boxEl = document.querySelector(".box");

console.log(boxEl);
```


불편한 진실
KBS2
왜 이러는 걸까요?

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
  <script src="./main.js"></script>
</head>
<body>
  <div class="box">Box!!</div>
</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <div class="box">Box!!</div>
  <script src="./main.js"></script>
</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
  <script defer src="./main.js"></script>
</head>
<body>
  <div class="box">Box!!</div>
</body>
</html>
```

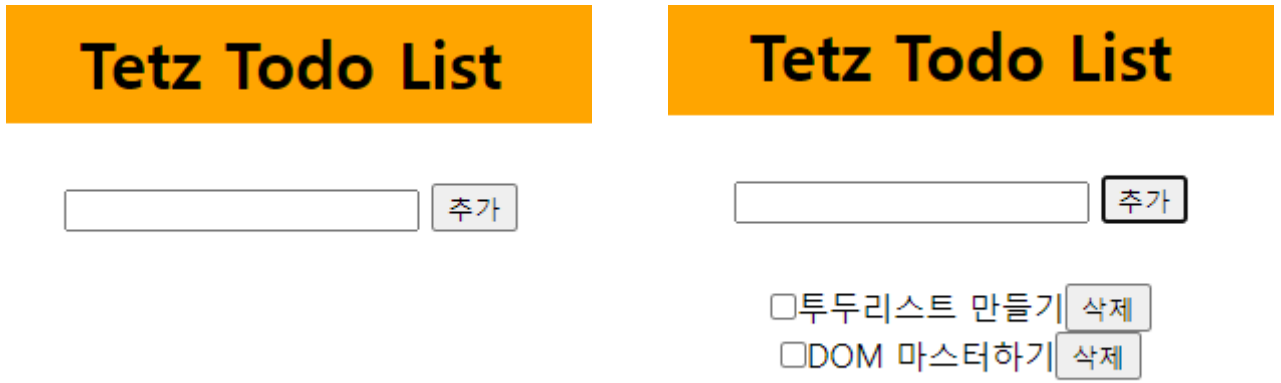

<script>
HTML parsing
Script fetch
Script execution
HTML parsing paused
<script async>
HTML parsing
Script fetch
Script execution
HTML parsing paused
<script defer>
HTML parsing
Script fetch
Script execution

TodoList

만들기!

자율 과제

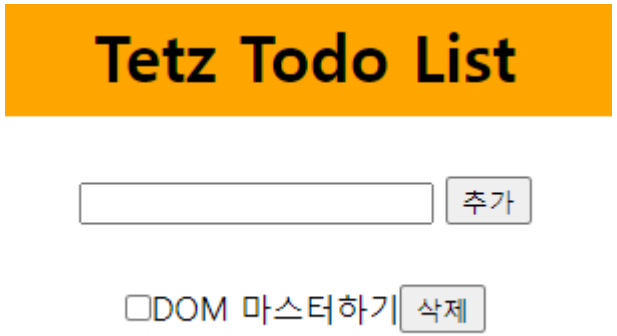
- 인풋에 내용을 입력하고 추가 버튼을 클릭하면 할 일이 추가 되는 웹 페이지를 만들어 봅시다!



자율 과제

- 할 일을 입력하지 않고 추가를 누르면 placeholder 에 내용을 입력하세요가 뜹니다!

자율 과제



- 체크 버튼을 체크하면 할 일 목록에 line-through 가 생깁니다
- 삭제 버튼을 클릭하면 해당 할 일 목록은 삭제 됩니다

힌트

- Check 박스가 check 되었는지 여부 체크

```
const checkBtn = document.createElement("input");
if (checkBtn.checked === true) {}
```

힌트

- 특정 요소를 먼저 붙이고 기능을 등록하는 방법은 어려우므로, 먼저 기능을 addEventListener 로 등록한 다음 붙이는 방법이 편합니다!

```
const checkBtn = document.createElement("input");
checkBtn.setAttribute("type", "checkbox");

checkBtn.addEventListener("click", function () {
  if (checkBtn.checked === true) {
    checkBtn.parentNode.style.textDecoration = "line-through";
  } else {
    checkBtn.parentNode.style.textDecoration = "none";
  }
});

addLi.append(checkBtn);
```


수고하셨습니다!