



Markup Language
Content



Style sheet Language
Presentation



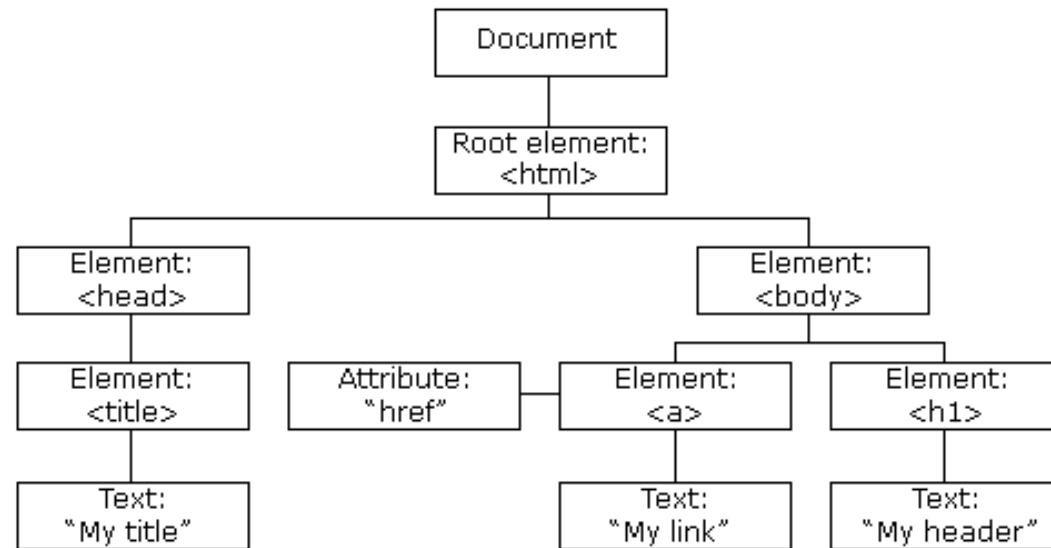
Programming Language
Behavior

자바스크립트 기본 문법 3

DOM

문서 객체 모델(DOM-Document Object Model)

- 자바스크립트가 웹 문서에 접근하고 제어할 수 있도록 객체를 사용해 웹 문서를 체계적으로 정리하는 방법
- 웹 페이지가 로드될 때 브라우저의 렌더링 엔진은 웹 문서를 브라우저가 이해할 수 있는 객체들의 트리구조로 구성(DOM 생성)하여 메모리에 적재



■ DOM 트리

- 웹 문서에 있는 요소들 간의 부모, 자식 관계를 계층 구조로 표시한 것


DOM 요소에 접근하기

■ getElementById() 메서드

- id 선택자로 접근
- 예) id값이 heading인 요소에 접근: `document.getElementById("heading")`

■ getElementByClassName() 메서드

- class 값으로 접근
- 예) `class="bright"` 속성이 있는 모든 요소를 찾으려면
 - `document.getElementsByClassName("bright")`

- 
- 반환 값이 2개 이상일 수 있음
 - `HTMLCollection` 객체에 저장됨

■ getElementByTagName() 메서드

- 태그 이름으로 접근
- 예) `<p>` 태그를 사용한 모든 요소에 접근
 - `document.getElementsByTagName("p")`

DOM 요소에 접근하기

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

HTML 내용 변경

■ 웹 요소의 내용을 수정하는 프로퍼티

- innerHTML : HTML 태그까지 포함해서 텍스트 내용 지정

```
<h1 id="id01">Old Heading</h1>

<script>
const element = document.getElementById("id01");
element.innerHTML = "New Heading";
</script>
```

■ 웹 요소의 속성의 값을 수정하는 프로퍼티

- document.getElementById(id).attribute = new value

```


<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>
```

자바스크립트 폼(Forms)

■ 입력 양식 빈칸 검사

```
function validateForm() {  
    let x = document.forms["myForm"]["fname"].value;  
    if (x == "") {  
        alert("Name must be filled out");  
        return false;  
    }  
}
```

```
<form name="myForm" action="/action_page.php" onsubmit="return validateForm()" method="post">  
Name: <input type="text" name="fname">  
<input type="submit" value="Submit">  
</form>
```

```
<form action="/action_page.php" method="post">  
    <input type="text" name="fname" required>  
    <input type="submit" value="Submit">  
</form>
```

자바스크립트 폼(Forms)

■ 숫자 범위 검사

JavaScript Validation

Please input a number between 1 and 10:

```
<h2>JavaScript Validation</h2>
<p>Please input a number between 1 and 10:</p>
<input id="numb">
<button type="button" onclick="myFunction()">Submit</button>
<p id="demo"></p>

<script>
function myFunction() {
    // Get the value of the input field with id="numb"
    let x = document.getElementById("numb").value;
    // If x is Not a Number or less than one or greater than 10
    let text;
    if (isNaN(x) || x < 1 || x > 10) {
        text = "Input not valid";
    } else {
        text = "Input OK";
    }
    document.getElementById("demo").innerHTML = text;
}
</script>
```

HTML 스타일 변경

- document.getElementById(id).style.*property* = new style

```
<p id="p2">Hello World!</p>
```

```
<script>
```

```
document.getElementById("p2").style.color = "blue";
```

```
</script>
```

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

예제) 도형의 테두리와 배경색 바꾸기

- 사각형 위에 마우스 포인터를 올려놓으면 초록색 원으로 바뀌고 내려놓으면 원래 도형으로 되돌아가는 예제

```
<style>
  #rect {
    width: 100px;
    height: 100px;
    background-color: red;
  }
</style>
```



```
<div id="rect"></div>
<script>
  let myRect = document.getElementById("rect");
  myRect.addEventListener("mouseover", function(){
    myRect.style.backgroundColor = "green";
    myRect.style.borderRadius="50%";
  });
  myRect.addEventListener("mouseout", function(){
    myRect.style.backgroundColor = "";
    myRect.style.borderRadius="";
  });
</script>
```

이벤트

태그 안에서 이벤트를 처리할 때는 "on"+"이벤트명" 사용 (예, 클릭하면 onclick 사용)

■ 마우스 이벤트

종류	설명
click	사용자가 HTML 요소를 클릭할 때 이벤트가 발생합니다.
dblclick	사용자가 HTML 요소를 더블클릭할 때 이벤트가 발생합니다.
mousedown	사용자가 요소 위에서 마우스 버튼을 눌렀을 때 이벤트가 발생합니다.
mousemove	사용자가 요소 위에서 마우스 포인터를 움직일 때 이벤트가 발생합니다.
mouseover	마우스 포인터가 요소 위로 옮겨질 때 이벤트가 발생합니다.
mouseout	마우스 포인터가 요소를 벗어날 때 이벤트가 발생합니다.
mouseup	사용자가 요소 위에 놓인 마우스 버튼에서 손을 뗄 때 이벤트가 발생합니다.

■ 키보드 이벤트

종류	설명
keydown	사용자가 키를 누르는 동안 이벤트가 발생합니다.
keypress	사용자가 키를 눌렀을 때 이벤트가 발생합니다.
keyup	사용자가 키에서 손을 뗄 때 이벤트가 발생합니다.

■ 문서 로딩 이벤트

종류	설명
abort	문서가 완전히 로딩되기 전에 불러오기를 멈췄을 때 이벤트가 발생합니다.
error	문서가 정확히 로딩되지 않았을 때 이벤트가 발생합니다.
load	문서 로딩이 끝나면 이벤트가 발생합니다.
resize	문서 화면 크기가 바뀌었을 때 이벤트가 발생합니다.
scroll	문서 화면이 스크롤되었을 때 이벤트가 발생합니다.
unload	문서에서 벗어날 때 이벤트가 발생합니다.

■ 폼 이벤트

종류	설명
blur	폼 요소에 포커스를 잃었을 때 이벤트가 발생합니다.
change	목록이나 체크 상태 등이 변경되면 이벤트가 발생합니다. <input>, <select>, <textarea> 태그에서 사용합니다.
focus	폼 요소에 포커스가 놓였을 때 이벤트가 발생합니다. <label>, <select>, <textarea>, <button> 태그에서 사용합니다.
reset	폼이 리셋되었을 때 이벤트가 발생합니다.
submit	submit 버튼을 클릭했을 때 이벤트가 발생합니다.

DOM 요소에 이벤트 처리기 함수 연결

■ onclick Event

- 클릭시 이벤트 연결

```
<h1 onclick="this.innerHTML = 'Oops!'">Click on this text!</h1>
```

```
<h1 onclick="changeText(this)">Click on this text!</h1>
<script>
function changeText(id) {
    id.innerHTML = "Oops!";
}
</script>
```

```
<script>
document.getElementById("myBtn").onclick = displayDate;
</script>
```

함수 이름 다음에 괄호 없음

버튼 클릭해서 설명 글 열고 닫기 - DOM 이벤트 처리기 연결

```
<div id="item">
  
  <button class="over" id="open">상세 설명 보기</button>
  <div id="desc" class="detail">
    <h4>등심붓꽃</h4>
    <p>북아메리카 원산으로 각지에서 관상초로 흔히 심고 있는 귀화식물</p>
    <button id="close">상세 설명 닫기</button>
  </div>
</div>

<script>
  document.querySelector('#open').onclick = function() {
    document.querySelector('#desc').style.display = "block";
    document.querySelector('#open').style.display = "none";
  }
  document.querySelector('#close').onclick = function() {
    document.querySelector('#desc').style.display = "none";
    document.querySelector('#open').style.display = "block";
  }
</script>
```

DOM 요소에 이벤트 처리기 함수 연결

■ onload, onunload Events

- 웹 페이지에 들어가거나 나갈 때 발생하는 이벤트

```
<body onload="checkCookies()">
```

■ oninput Event

- 사용자 데이터를 입력하는 동안에 발생

```
<input type="text" id="fname" oninput="upperCase()">
```

```
<script>
function upperCase() {
  const x = document.getElementById("fname");
  x.value = x.value.toUpperCase();
}
</script>
```

Enter your name:

DOM 요소에 이벤트 처리기 함수 연결

■ onchange Event

- 입력 필드에서 입력이 끝나고 다른 곳으로 포커스가 변경될 때 발생
- 즉, 입력되는 동안은 소문자로 지속되다가 포커스 변경될 때 대문자로 변경

Enter your name:

```
<input type="text" id="fname" onchange="upperCase()">
```

■ onmouseover, onmouseout Event

■ onmousedown, onmouseup

```
<div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background-color:#D94A38;width:120px;height:20px;padding:40px;">
Mouse Over Me</div>
```

```
<script>
function mOver(obj) {
    obj.innerHTML = "Thank You"
}

function mOut(obj) {
    obj.innerHTML = "Mouse Over Me"
}
</script>
```

```
<div onmousedown="mDown(this)" onmouseup="mUp(this)"
style="background-color:#D94A38;width:90px;height:20px;padding:40px;">
Click Me</div>
```

```
<script>
function mDown(obj) {
    obj.style.backgroundColor = "#1ec5e5";
    obj.innerHTML = "Release Me";
}

function mUp(obj) {
    obj.style.backgroundColor="#D94A38";
    obj.innerHTML="Thank You";
}
</script>
```

addEventListener() 메소드 사용하기

■ 요소.addEventListener(이벤트, 함수);

- 이벤트는 "on" prefix를 사용하지 않음. onclick이 아닌 click 만 작성

```
element.addEventListener("click", function(){ alert("Hello World!"); });
```

```
element.addEventListener("click", myFunction);
```

```
function myFunction() {  
    alert ("Hello World!");  
}
```

■ 요소.removeEventListener("click", myfunction);

- 이벤트 핸들러 제거

라이트 박스 효과 만들기

■ 작동 흐름

- 섬네일 이미지 6개를 화면에 보여줍니다.
- 섬네일 이미지 가운데 하나를 클릭하면 그 이미지를 라이트 박스 영역에 표시하고 화면에 나타나게 합니다.
- 화면에 나타난 라이트 박스 영역을 클릭하면 다시 라이트 박스를 감춥니다.

■ lightbox.html

```
<div class="row">
  <ul>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
  </ul>
</div>
<div id="lightbox">
  
</div>
```


CSS

```
.row {
    width:420px;
    margin:0 auto;
}

.row ul {
    list-style:none;
    margin:0;
    padding:0;
}

.row ul li {
    display:inline-table;
}

/* 라이트 박스 스타일 */
#lightbox {
    position: fixed; /* 위치 고정 */
    width:100%; /* 너비 */
    height:100%; /* 높이 */
    background-color: rgba(0,0,0,0.7); /* 배경색 */
    top:0; /* 시작 위치 - 위쪽 끝 */
    left:0; /* 시작 위치 - 왼쪽 끝 */
    display:none; /* 화면에서 감추기 */
}

/* 라이트 박스 안의 이미지 */
#lightbox img {
    position:absolute; /* top, left에 의해 위치 지정 */
    top:50%; /* 위쪽에서 50% 부터 */
    left:50%; /* 왼쪽에서 50% 부터 */
    transform:translate(-50%, -50%); /* 요소를 화면 중앙에 표시하기 위해 이동 */
    border:5px solid #eee; /* 이미지 테두리 */
}
```

라이트 박스 효과 만들기

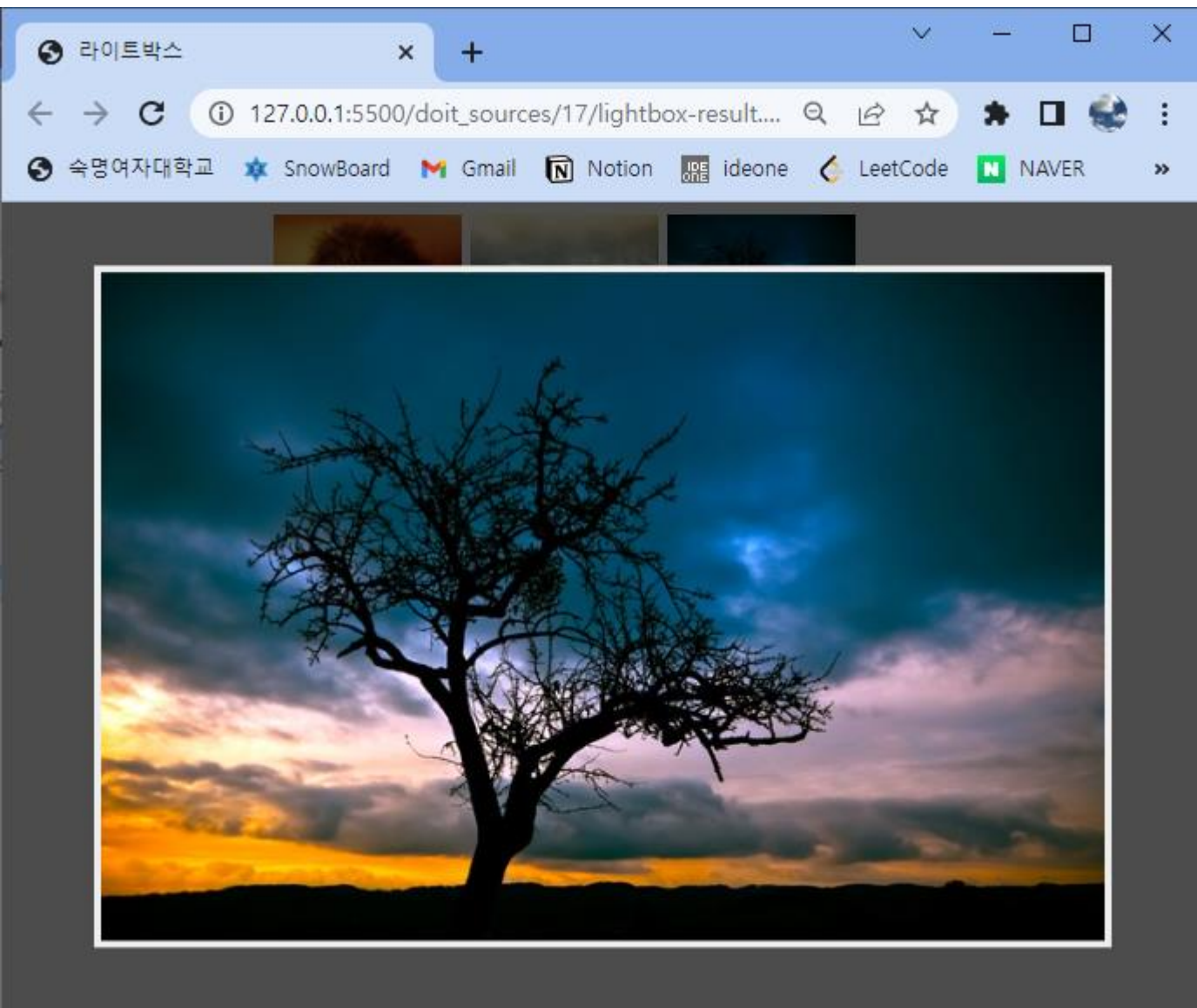
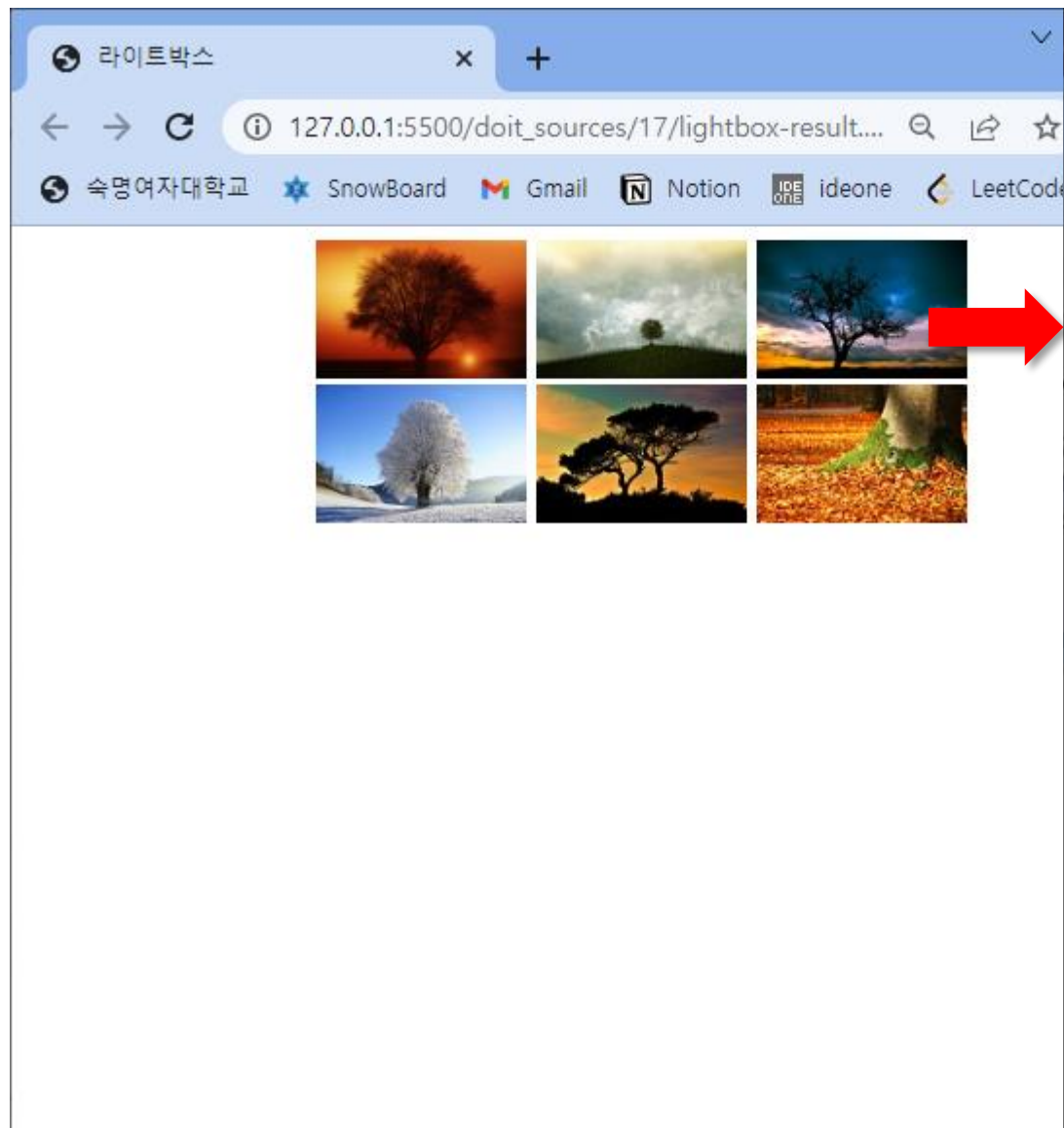
```
<script>
  var pics = document.getElementsByClassName("pic"); // .pic인 요소들을 가져와 pics 라는 변수에 저장
  var lightbox = document.getElementById("lightbox"); // 라이트 박스. querySelector("#lightbox")
  var lightboxImage = document.getElementById("lightboxImage"); // 라이트 박스 안의 이미지

  for (i=0; i<pics.length; i++) {
    pics[i].addEventListener("click", showLightbox);
  }

  function showLightbox() {
    var bigLocation = this.getAttribute("data-src"); // bigLocation = this.data-src; 도 가능
    lightboxImage.setAttribute("src", bigLocation); // lightboxImage.src = bigLocation;
    lightbox.style.display = "block"; // 라이트박스 이미지를 화면에 표시
  }

  lightbox.onclick = function() { //click 이벤트가 발생했을 때 실행할 함수 선언
    lightbox.style.display = "none"; // lightbox 요소를 화면에서 감춤
  }
</script>
```

라이트 박스 효과 만들기



DOM에서 노드 추가, 삭제하기

- DOM에서 새로운 노드를 만들어 추가하거나 삭제하려면 노드 리스트를 사용해야 함.
- 노드 리스트(node list)란
 - querySelectorAll() 메서드를 사용해 가져온 여러 개의 노드를 저장한 것

(... 생략 ...)

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



새로운 요소 추가하기

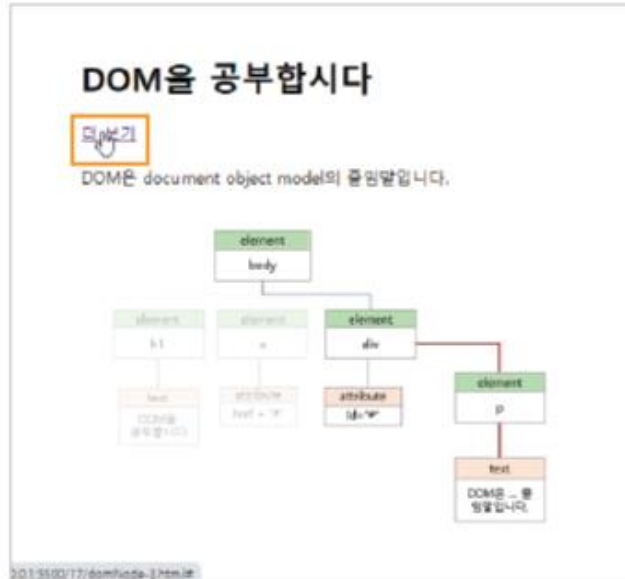
- 1. 요소 노드 만들기 - createElement()
- 2. 텍스트 노드 만들기 - createTextNode()
- 3. 자식 노드 연결하기 - appendChild()

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
const para = document.createElement("p");
const node = document.createTextNode("This is new.");
para.appendChild(node);
document.getElementById("div1").appendChild(para);
</script>
```

속성 값이 있는 새로운 요소 추가하기

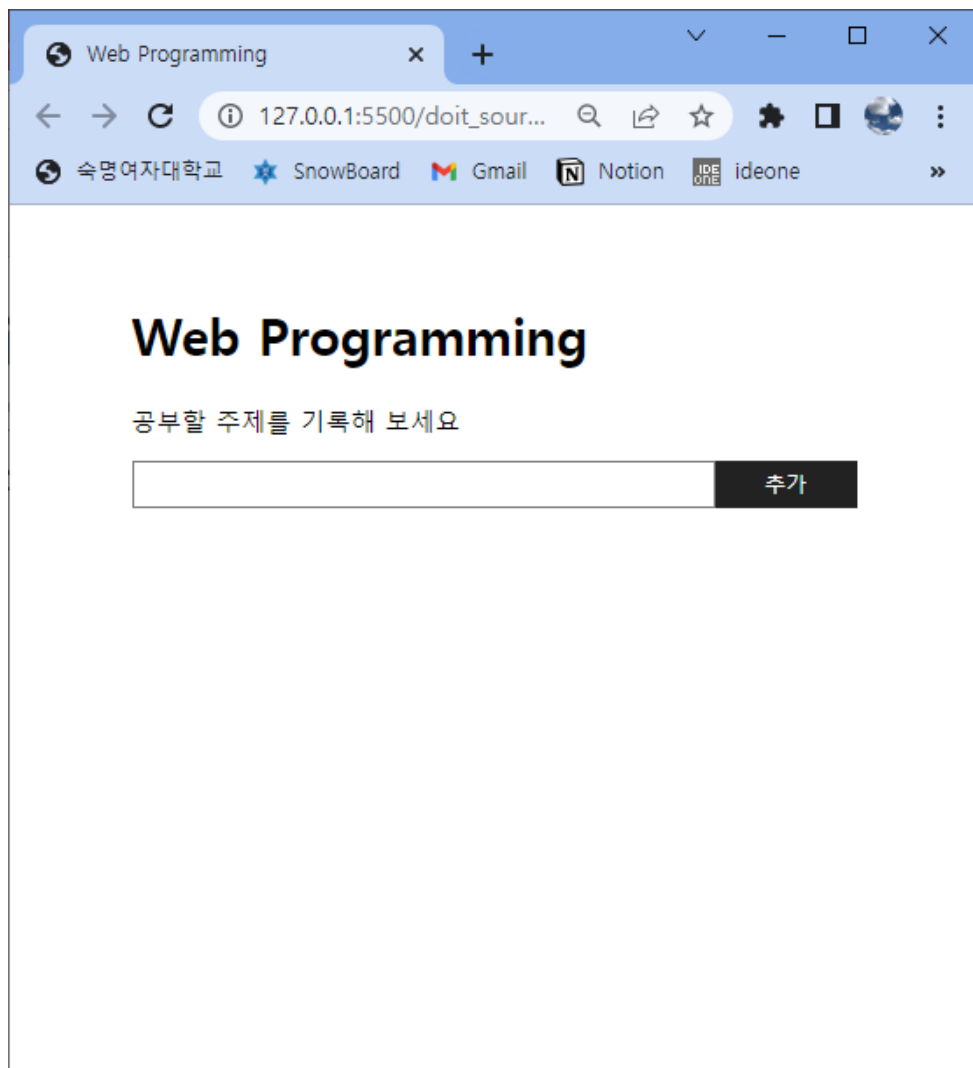
- 1. 요소 노드 만들기 - createElement() 메서드
- 2. 속성 노드 만들기 - createAttribute() 메서드
- 3. 속성 노드 연결하기 - setAttributeNode() 메서드
- 4. 자식 노드 연결하기 - appendChild() 메서드



```
(... 생략 ...)  
<div id="container">  
  <h1>DOM을 공부합시다</h1>  
  <a href="#" onclick="addContents(); this.onclick='';">더 보기</a>  
  <div id="info"></div>  
</div>  
<script>  
  function addContents() {  
    var newP = document.createElement("p");  
    var txtNode = document.createTextNode("DOM은 document object model의 줄임말입  
니다.");  
    newP.appendChild(txtNode);  
  
    var newImg = document.createElement("img");  
    var srcNode = document.createAttribute("src");  
    var altNode = document.createAttribute("alt");  
    srcNode.value = "images/dom.jpg";  
    altNode.value = "돔 트리 예제 이미지";  
    newImg.setAttributeNode(srcNode);  
    newImg.setAttributeNode(altNode);  
  
    document.getElementById("info").appendChild(newP);  
    document.getElementById("info").appendChild(newImg);  
  }  
</script>  
(... 생략 ...)
```

앞의 예제에서 추가된 부분입니다.

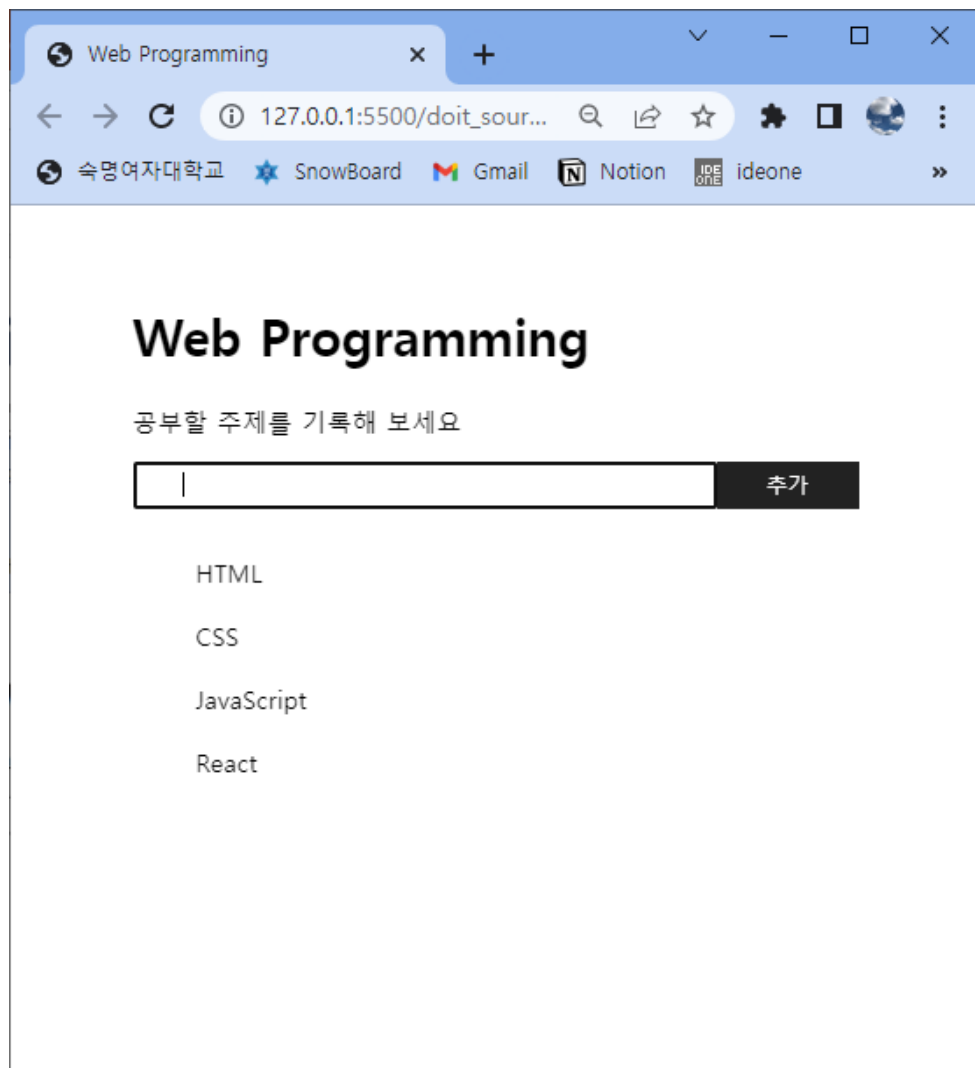
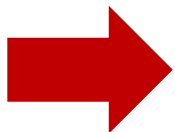
예제> 텍스트 필드에 입력한 값을 화면에 표시하기



Web Programming

공부할 주제를 기록해 보세요

추가



Web Programming

공부할 주제를 기록해 보세요

추가

- HTML
- CSS
- JavaScript
- React

예제> 텍스트 필드에 입력한 값을 화면에 표시하기

```
<div id="container">
  <h1>Web Programming</h1>
  <p>공부할 주제를 기록해 보세요</p>
  <form action="">
    <input type="text" id="subject" autofocus>
    <button onclick="newRegister(); return false;">추가</button>
  </form>
  <hr>
  <ul id="itemList"></ul>
</div>
<script>
  function newRegister() {
    var newItem = document.createElement("li"); // 요소 노드 추가
    var subject = document.querySelector("#subject"); // 폼의 텍스트 필드
    var newText = document.createTextNode(subject.value); // 텍스트 필드의 값을 텍스트 노드로 만들기
    newItem.appendChild(newText); // 텍스트 노드를 요소 노드의 자식 노드로 추가

    var itemList = document.querySelector("#itemList"); // 웹 문서에서 부모 노드 가져오기
    itemList.appendChild(newItem); // 새로 만든 요소 노드를 부모 노드에 추가

    subject.value="";
  }
</script>
```

기본 appendChild → 새로운 노드를 부모 노드의 맨 끝에 추가

예제> 텍스트 필드에 입력한 값을 화면에 표시하기

- `itemList.appendChild(newItem);` // 자식 노드를 맨 마지막에 추가
- `itemList.insertBefore(newItem, itemList.childNodes[0]);` // 자식 노드를 맨 앞에 추가

```
<script>
function newRegister() {
    var newItem = document.createElement("li"); // 요소 노드 추가
    var subject = document.querySelector("#subject"); // 폼의 텍스트 필드
    var newText = document.createTextNode(subject.value); // 텍스트 필드의 값을 텍스트 노드로 만들기
    newItem.appendChild(newText); // 텍스트 노드를 요소 노드의 자식 노드로 추가

    var itemList = document.querySelector("#itemList"); // 웹 문서에서 부모 노드 가져오기
    itemList.insertBefore(newItem, itemList.childNodes[0]); // 자식 노드중 첫번째 노드 앞에 추가
    // itemList.appendChild(newItem);

    subject.value="";
}
</script>
```

노드 삭제하기

■ removeChild() 메소드

- 자식 노드 삭제
- 형식: 부모노드.removeChild(자식노드)
- 노드를 삭제할 때는 부모 노드에서 자식 노드를 삭제해야 함 (old version)
- 즉, 노드를 삭제하려면 부모 노드부터 찾아야 함

■ parentNode 프로퍼티

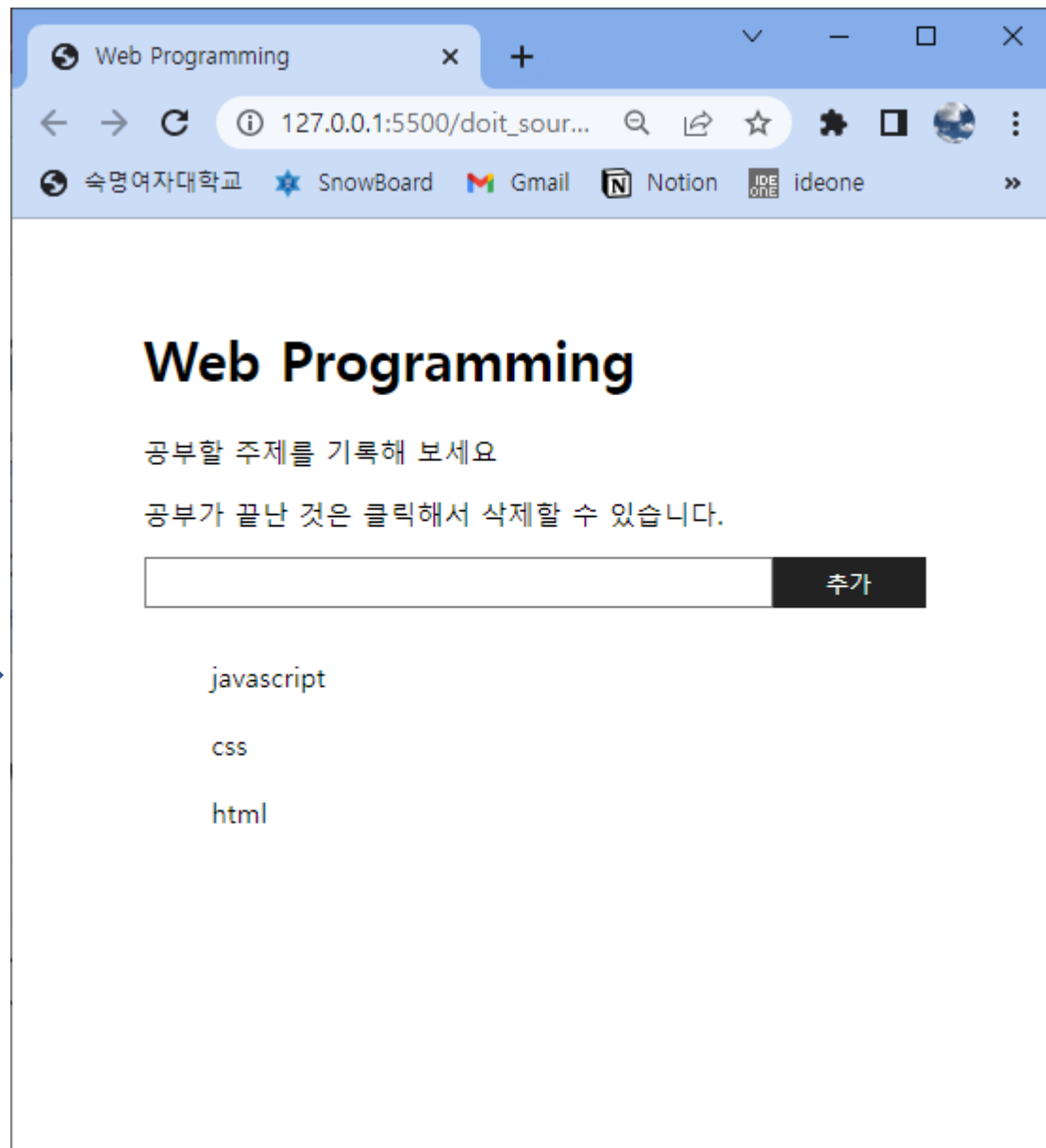
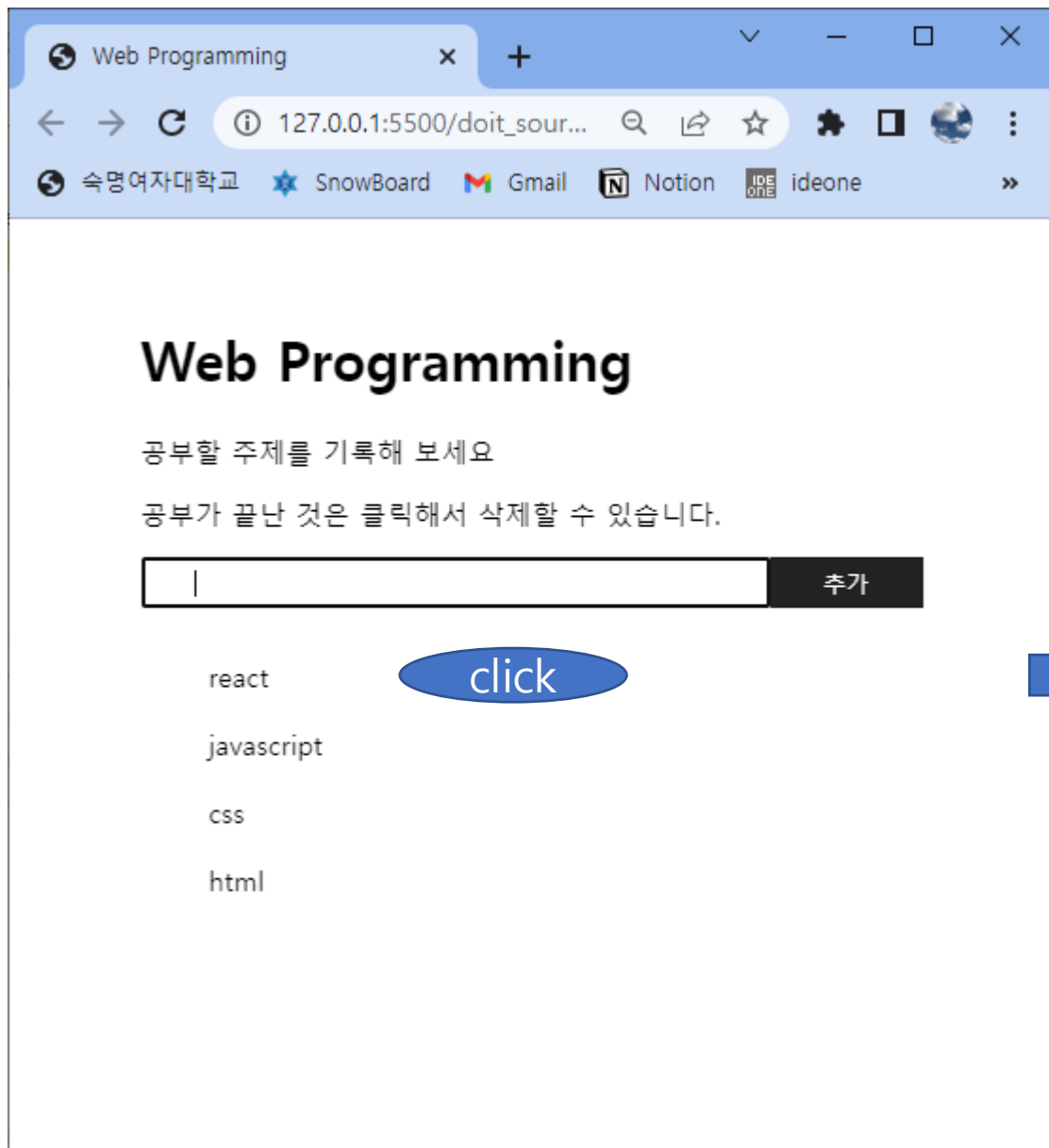
- 현재 노드의 부모 노드에 접근해서 부모 노드의 요소 노드를 반환
- 형식: 노드.parentNode

■ remove() 메소드 (old version 브라우저에서는 지원하지 않음)

- 해당 노드 삭제

```
document.getElementById("p1").remove();
```

예제> 입력한 항목을 클릭하여 삭제하기



예제> 입력한 항목을 클릭하여 삭제하기

```
<script>
function newRegister() {
    var newItem = document.createElement("li"); // 요소 노드 추가
    var subject = document.querySelector("#subject"); // 폼의 텍스트 필드
    var newText = document.createTextNode(subject.value); // 텍스트 필드의 값을 텍스트 노드로 만들기
    newItem.appendChild(newText); // 텍스트 노드를 요소 노드의 자식 노드로 추가

    var itemList = document.querySelector("#itemList"); // 웹 문서에서 부모 노드 가져오기
    itemList.insertBefore(newItem, itemList.childNodes[0]); // 자식 노드중 첫번째 노드 앞에 추가

    subject.value="";

    var items = document.querySelectorAll("li"); // 모든 항목 가져오기
    for(i=0; i<items.length; i++) {
        items[i].addEventListener("click", function() { // 항목 클릭했을 때 실행할 함수
            if(this.parentNode) // 부모 노드가 있다면
                this.parentNode.removeChild(this); // 부모 노드에서 삭제
        });
    }
}
</script>
```