Javascript 기초

이벤트 처리 및 DOM 조작

김태민

저번에 우리는

객체

객체라는 용어의 범위는 자바스크립트에서 매우 포괄적
자료형의 관점에서 보면 키(key)와 값(value)으로 구성된 속성의 집합
객체는 {}를 이용해 생성할 수 있는데, 이런 방법을 리터럴(literal) 방식으로 객체를 생성했다고 표현

```
const person = {};
```

속성이 한 개도 없는 객체를 빈 객체라고 합니다.

사람에 대한 객체를 생성한다면 다음과 같은 속성을 지정해 객체를 생성할 수 있음

```
const person = {"Hong Gildong"};
```

배열

JavaScript 배열

배열(Array)은 여러 값을 순서대로 저장하는 데이터 구조

각 값은 인덱스(0부터 시작하는 숫자)로 접근

동적 크기: 필요에 따라 길이 조정 가능

다양한 데이터 타입 저장 가능(숫자, 문자열, 객체 등)

웹 개발에서 데이터 목록(예: 제품 목록, 사용자 리스트) 관리에 필수

https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array

실습 환경

실습환경

OS : Window / Mac

브라우저 : Chrome

에디터 : VS Code https://code.visualstudio.com/

VS Code 익스텐션 : ESLint, Prettier, HTML CSS Support, HTML to CSS autocompletion, Auto Rename Tag, Auto Close Tag, htmltagwrap

Git : git bash, github 가입

DOM

DOM (Document Object Model) 문서객체모델

웹 브라우저는 HTML 문서의 구성 요소를 모두 객체로 인식
웹 브라우저는 문서 객체 모델(DOM)을 생성할 수 있음
문서 객체 모델이란 웹 브라우저에 표시되는 HTML 문서 구조를 객체화한 모델
웹 브라우저는 생성한 문서 객체 모델을 통해 HTML 문서의 구성 요소를 객체로 인식

자바스크립트도 웹 브라우저의 문서 객체 모델을 조작해
웹 브라우저에 표시되는 HTML 문서 구조를 **변경**하거나 새로운 구성 요소를 **추가**하는 등의 작업 가능

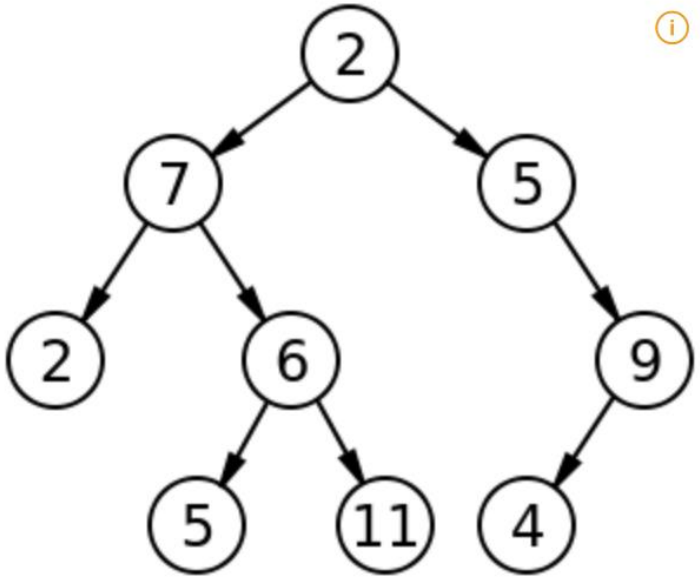
DOM (Document Object Model) 문서객체모델

DOM은 웹 브라우저가 HTML 문서를 해석하고, 해석한 HTML 문서 구조를 객체로 변환하는 방식으로 생성

웹 브라우저에 표시되는 HTML 문서는 내부적으로 문서 객체 모델을 해석

문서 객체 모델은 다음 그림처럼 나무를 뒤집어 놓은 형태의 자료구조인 트리(tree) 구조를 가짐.

그래서 이를 가리켜 DOM 트리라고 부름

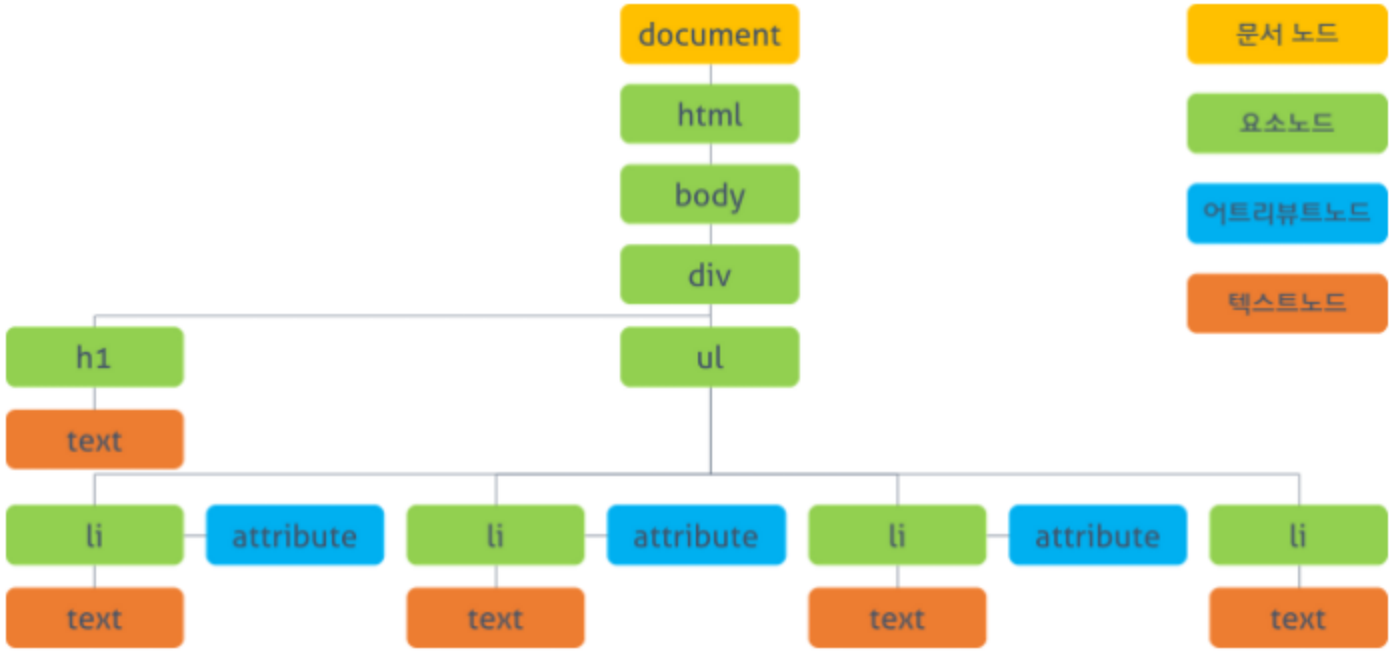


DOM

DOM (Document Object Model) 문서객체모델

가장 꼭대기(최상위)에 있는 노드를 루트 노드(root node)

document는 노드가 아니라 객체이므로 여기서는 html을 루트 노드라 부름



노드

자바스크립트로 웹 브라우저에 표시되는 HTML 문서를 조작하려면 문서 객체 모델을 조작해야 함
문서 객체 모델은 window 객체의 document 객체를 사용해 조작할 수 있음.

자바스크립트로 HTML 문서를 조작할 때 가장 먼저 할 일은
document 객체로 조작하려고 하는 문서 객체 모델의 노드를 선택하는 것

노드는 문서 노드부터 주석 노드까지 여러 타입이 있지만,
자바스크립트는 주로 요소 노드(html, body, div, p ...)를 선택해 조작합니다.

노드

모든 노드 탐색	parentNode	부모 노드를 반환합니다.
	childNodes	모든 자식 노드를 반환합니다.
	firstChild	첫 번째 자식 노드를 반환합니다.
	lastChild	마지막 자식 노드를 반환합니다.
	previousSibling	이전 형제 노드를 반환합니다.
	nextSibling	다음 형제 노드를 반환합니다.

document.firstChild; // <DOCTYPE html>

document.firstElementChild; // html

요소 노드 탐색	parentElement	부모 요소 노드를 반환합니다.
	children	자식 요소 노드를 반환합니다.
	firstElementChild	첫 번째 자식 요소 노드를 반환합니다.
	lastElementChild	마지막 자식 요소 노드를 반환합니다.
	previousElementSibling	이전 요소 노드를 반환합니다.
	nextElementSibling	다음 요소 노드를 반환합니다.

속성값과 태그명으로 노드 선택하기 - get 메서드

메서드 형식	설명
getElementById(<id 속성값>)	id 속성값과 일치하는 요소 노드를 1개만 선택합니다.
getElementsByClassName(<class 속성값>)	class 속성값과 일치하는 요소 노드를 모두 선택합니다.
getElemenetsByTagName(<태그명>)	태그명과 일치하는 요소 노드를 모두 선택합니다.

DOM


```
<body>
    <h1 id="title">title</h1>
    <p class="text">text-1</p>
    <p class="text">text-2</p>
    <script>
        // id 속성값이 title인 요소 노드 1개 선택하기
        const el = document.getElementById("title");
        console.log(el);
        // class 속성값이 text인 요소 노드 모두 선택하기
        const classEl = document.getElementsByClassName("text");
        console.log(classEl);
        // p 태그에 해당하는 요소 노드 모두 선택하기
        const tagEls = document.getElementsByName("p");
        console.log(tagEls);
    </script>
</body>
```

```
<script>
    const classEl = document.getElementsByClassName("text");
    console.log(classEl[0]);
    console.log(classEl[1]);
    const tagEls = document.getElementsByTagName("p");
    console.log(tagEls[0]);
    console.log(tagEls[1]);
</script>
```

속성값과 태그명으로 노드 선택하기 - get 메서드

메서드 형식	설명
querySelector(<CSS 선택자>)	매개변수로 넘어오는 CSS 선택자에 해당하는 노드를 1개만 선택합니다.
querySelectorAll(<CSS 선택자>)	매개변수로 넘어오는 CSS 선택자에 해당하는 노드를 모두 선택합니다.

```
<body>
    <div class="box-1">
        <p class="text">text-1</p>
        <p class="text">text-2</p>
    </div>
    <div class="box-2">
        <p class="text">text-3</p>
        <p class="text">text-4</p>
    </div>
</body>
```

```
const el = document.getElementsByClassName("box-1")[0].children;
console.log(el);
```

```
const el = document.querySelectorAll(".box-1 .text");
console.log(el);
```

노드 조작

콘텐츠 조작하기

속성	설명
textContent	요소 노드의 모든 텍스트에 접근합니다.
innerText	요소 노드의 텍스트 중 웹 브라우저에 표시되는 텍스트에만 접근합니다.
innerHTML	요소 노드의 텍스트 중 HTML 태그를 포함한 텍스트에만 접근합니다.

```
<p id="title">Hello, <span style="display:none;">Javascript!</span></p>
```

```
document.getElementById("title").textContent; // Hello, Javascript!
document.getElementById("title").innerText;   // Hello,
document.getElementById("title").innerHTML;   // Hello, <span style="display: none;">Javascript!</span>
```

콘텐츠 조작하기

```
<p id="textContent"></p>
<p id="innerText"></p>
<p id="innerHTML"></p>
<script>
    document.querySelector("#textContent").textContent = `<strong>textContent</strong> 속성`;
    document.querySelector("#innerText").innerText = `<strong>innerText</strong> 속성`;
    document.querySelector("#innerHTML").innerHTML = `<strong>innerHTML</strong> 속성`;
</script>
```

노드 조작

스타일 조작하기

<노드>.style.<css 속성명> = <속성값>;

```
<p id="text">text</p>
<script>
    const pEl = document.querySelector("p"); // 노드 선택하기
    pEl.style.color = "red";
</script>
```

노드 조작

스타일 조작하기

querySelector() 메서드로 스타일을 조작하고 싶은 노드를 선택

선택한 노드에 style 속성으로 조작하고 싶은 CSS 속성명을 적고, 적용하고 싶은 CSS 속성값을 할당

카멜 표기법으로 작성 : background-color => backgroundColor

```
<p id="text">text</p>
<script>
    const pEl = document.querySelector("p"); // 노드 선택하기
    pEl.style.backgroundColor = "#ff0000";
    pEl.style.fontSize = "20px";
    pEl.style.color = "#ffffff";
</script>
```

클래스 속성 조작하기

Style 속성으로 조작하려면 불편함

```
<body>
    <p>text</p>
    <script>
        const pEl = document.querySelector("p");
        pEl.style.color = "red";
        pEl.style.fontSize = "20px";
        pEl.style.fontWeight = "bold";
        pEl.style.lineHeight = "1";
    </script>
</body>
```

```
<style>
    .active{
        color:red;
        font-size:20px;
        font-weight:bold;
        line-height:1;
    }
</style>
</head>
<body>
    <p class="active">text</p>
</body>
```

노드 조작

클래스 속성 조작하기

선택한 요소 노드에 class 속성을 지정할 때는 classList 속성의 add(), remove(), toggle() 사용

```
// 추가
<노드>.classList.add("class 속성값");
// 삭제
<노드>.classList.remove("class 속성값");
// 추가와 삭제 반복
<노드>.classList.toggle("class 속성값");
```

```
<style>
    .red-color{
        color:red;
    }
    .fz20{
        font-size:20px;
    }
</style>
<p id="text">text</p>
<script>
    const pEl = document.querySelector("#text"); // 노드 선택하기
    pEl.classList.add("red-color");
    pEl.classList.add("fz20");
</script>
```

클래스 속성 조작하기

class 속성을 한 번에 추가

```
<script>
    const pEl = document.querySelector("#text");
    pEl.classList.add("red-color", "fz20");
</script>
```

클래스 속성 조작하기

적용된 class 속성값을 삭제하고 싶을 때는 classList 속성의 remove() 메서드를 사용

```
<p id="text" class="red-color fz20">text</p>
<script>
    const pEl = document.querySelector("#text");
    pEl.classList.remove("red-color", "fz20"); // 삭제
</script>
```

클래스 속성 조작하기

toggle() 메서드는 add() 메서드와 remove() 메서드를 반복해서 호출

```
<p id="text">text</p>
<script>
    const pEl = document.querySelector("#text"); // 노드 선택
    // 1초마다 toggle() 메서드 반복 실행
    setInterval(function(){
        pEl.classList.toggle("red-color");
    }, 1000);
</script>
```

데이터 속성 조작하기

HTML5에서 새로 추가된 data-* 속성. == 사용자 정의(custom) 속성.

HTML 문법에서 사용할 수 있는 속성 외에 사용자가 원하는 속성을 추가할 수 있음

```
<button data-cnt="10">가방 구매</button>
<button data-cnt="0">신발 구매</button>
<script>
    const buttonEls = document.querySelectorAll("button");
    buttonEls.forEach((el) => {
        console.log(el.dataset);
    })
</script>
```

데이터 속성 조작하기

dataset 속성으로 노드의 data-* 속성에 대한 정보를 DOMStringMap 객체에 담겨 반환됨

이 중에서 정확하게 data-cnt 속성의 값만 가져오고 싶으면 다음처럼 객체 속성에 접근하는 방법을 사용

```
<script>
    const buttonEls = document.querySelectorAll("button");
    buttonEls.forEach((el) => {
        console.log(el.dataset.cnt);
    })
</script>
```

데이터 속성 조작하기

속성에 값을 할당하면 단순하게 값을 가져오는 것이 아니라 data-cnt 속성의 값을 바꿀 수 있음

```
<script>
    const buttonEls = document.querySelectorAll("button");
    buttonEls.forEach((el) => {
        el.dataset.cnt = 50;
    })
</script>
```

데이터 속성 조작하기

지금까지 document 객체의 속성으로 HTML 요소에 접근해 일부 속성을 조작

다음 메서드를 사용하면 모든 속성을 전체적으로 조작할 수 있음

메서드 형식	설명
<노드>.getAttribute("속성명");	속성값을 가져옵니다.
<노드>.setAttribute("속성명", "속성값");	속성값을 설정합니다.
<노드>.removeAttribute("속성명");	속성을 삭제합니다.

데이터 속성 조작하기

getAttribute() 메서드는 선택된 요소 노드의 속성값을 가져오고 싶을 때 사용

querySelector() 메서드로 a 태그에 해당하는 요소 노드를 선택

getAttribute() 메서드로 href 속성값을 가져옴

```
<a href="https://ozcodingschool.com">OZ코딩</a>
<script>
    const aEl = document.querySelector("a");
    const href = aEl.getAttribute("href");
    console.log(href);
</script>
```

데이터 속성 조작하기

속성값을 새로 설정하고 싶을 때는 setAttribute() 메서드 사용

setAttribute() 메서드로 href 속성값을 새로 설정

a 태그의 텍스트를 바꾸기 위해 innerText 속성도 같이 사용

```
<a href="https://ozcodingschool.com">OZ코딩</a>
<script>
    const aEl = document.querySelector("a");
    const href = aEl.getAttribute("href");
    aEl.setAttribute("href", "https://goosepeak.kr");
    aEl.innerText = "구스피크";
</script>
```

데이터 속성 조작하기

getAttribute() 메서드와 setAttribute() 메서드는 모든 속성의 상위 메서드

classList 속성이나 dataset 속성으로 하는 조작을 전부 할 수 있음

```
<style>
    .red-color{ color:red; }
</style>
<a href="https://ozcodingschool.com">OZ코딩</a>
<script>
    const aEl = document.querySelector("a");
    aEl.setAttribute("data-link", " https://goosepeak.kr");
    aEl.setAttribute("class", "red-color");
    aEl.innerText = "구스피크";
</script>
```

데이터 속성 조작하기

removeAttribute() 메서드를 사용하면 요소 노드의 속성 제거

```
<a href="https://ozcodingschool.com">OZ코딩</a>
<script>
    const aEl = document.querySelector("a");
    aEl.removeAttribute("class"); // class 속성 삭제
</script>
```

노드 추가/삭제

노드 추가

구분	메서드	설명
노드 생성	createElement()	요소 노드를 생성합니다.
	createTextNode()	텍스트 노드를 생성합니다.
	createAttribute()	속성 노드를 생성합니다.
노드 연결	<기준 노드>.appendChild(<자식 노드>)	기준 노드에 자식 노드를 연결합니다.
	<기준 노드>.setAttributeNode(<속성 노드>)	기준 노드에 속성 노드를 연결합니다.

```
<!DOCTYPE html>
<html>
<head>
    <title>Create Node</title>
</head>
<body>
    <script></script>
</body>
</html>
```

노드 추가/삭제

노드 추가

```
<script>
    const aEl = document.createElement("a"); // 요소 노드 생성하기. 아직 dom에 붙이지 않음
    document.body.appendChild(aEl); // body에 자식요소로 붙이기

    const txtEl = document.createTextNode("OZ코딩");
    document.querySelector("a").appendChild(txtEl); // textNode를 a태그 자식요소로 붙이기

    const hrefAttr = document.createAttribute("href");
    hrefAttr.value = "https://www.ozcodingschool.com";
    document.querySelector("a").setAttributeNode(hrefAttr);  // href 속성 노드 추가하기
</script>
```

노드 추가/삭제

노드 삭제

```
<부모 노드>.removeChild(<자식 노드>)
```

```
<body>
    <p>text 1</p>
    <a href="https://www.ozcodingschool.com">OZ코딩</a>
    <a href="https://www.goosepeak.kr">구스피크</a>
    <script>
        const pEl = document.querySelector("p");
        pEl.parentNode.removeChild(pEl); // p태그 삭제
    </script>
</body>
```

노드 추가/삭제

노드 삭제

```
<body>
    <p>text 1</p>
    <a href="https://www.ozcodingschool.com">OZ코딩</a>
    <a href="https://www.goosepeak.kr">구스피크</a>
    <script>
        const childNodes = document.body.childNodes;
        childNodes.forEach((node) => {
        if(node.nodeName === "a")
            node.parentNode.removeChild(node);
        })
    </script>
</body>
```

이벤트 리스너

이벤트 리스너

주요 이벤트데이터 속성 조작하기

웹 브라우저에서 사용자와의 상호작용으로 발생하는 이벤트는 200여 가지가 넘음

구분	이벤트	설명
마우스 이벤트	onclick	마우스로 클릭하면 발생합니다.
	ondblclick	마우스로 빠르게 두 번 클릭하면 발생합니다.
	onmouseover	마우스 포인터를 올리면 발생합니다.
	onmouseout	마우스 포인터가 빠져나가면 발생합니다.
	onmousemove	마우스 포인터가 움직이면 발생합니다.
	onwheel	마우스 휠(wheel)을 움직이면 발생합니다.
키보드 이벤트	onkeypress	키보드 버튼을 누르고 있는 동안 발생합니다.
	onkeydown	키보드 버튼을 누른 순간 발생합니다.
	onkeyup	키보드 버튼을 눌렀다가 뗀 순간 발생합니다.
포커스 이벤트	onfocus	요소에 포커스가 되면 발생합니다.
	onblur	요소가 포커스를 잃으면 발생합니다.
폼 이벤트	onsubmit	폼이 전송될 때 발생합니다.

이벤트 리스너

이벤트 등록 - 인라인 방식

인라인 방식은 HTML 태그에 속성으로 이벤트를 등록하는 방법

onclick 이벤트를 button 태그의 속성으로 사용

속성값으로는 이벤트가 발생할 때 실행될 함수를 지정

버튼을 클릭해 보면 clickEvent() 함수가 실행되어 웹 브라우저에 경고창이 출력됨.

```
<button onclick="clickEvent()">클릭</button>
<script>
    function clickEvent(){
        alert("click");
    }
</script>
```

이벤트 리스너

이벤트 등록 - 인라인 방식

입력창을 클릭해서 커서를 활성화하면 onfocus 이벤트가 발생

입력창 외부 영역을 클릭하면 onblur 이벤트가 발생해 커서가 빠져나가면서 블러(focus out) 상태가 됨

```
<form>
    <input type="text" onfocus="focusEvent()" onblur="blurEvent()">
</form>
<script>
    function focusEvent(){
        console.log("focus on");
    }
    function blurEvent(){
        console.log("focus out");
    }
</script>
```

이벤트 리스너

이벤트 등록 – 프로퍼티 리스너 방식

요소 노드에 직접 속성으로 이벤트를 등록하는 방법

버튼을 클릭했을 때 요소 노드에 등록된 이벤트 속성에 할당된 함수가 실행됨

```
<button>클릭</button>
<script>
    const btnEl = document.querySelector("button");
    btnEl.onclick = function(){
        alert("click");
    }
</script>
```

이벤트 리스너

이벤트 등록 – 프로퍼티 리스너 방식

화살표 함수 사용 가능

```
<button>클릭</button>
<script>
    const btnEl = document.querySelector("button");
    btnEl.onclick = () => {
        alert('arrow click');
    }
</script>
```

이벤트 리스너

이벤트 등록 – 프로퍼티 리스너 방식

함수는 함수 선언문이나 함수 표현식, 화살표 함수 등 어떤 방식으로 정의해도 상관없음

```
<button>클릭</button>
<script>
    const btnEl = document.querySelector("button");
    btnEl.onclick = clickEvent;
    function clickEvent(){
        alert('click');
    }
</script>
```

이벤트 리스너

이벤트 등록 – 이벤트 등록 메서드 방식

DOM에서 제공하는 addEventListener() 메서드를 사용

```
<노드>.addEventListener("<이벤트 타입>", <이벤트 함수>);
```

```
<button>클릭</button>
<script>
    const btnEl = document.querySelector("button");
    btnEl.addEventListener("click", function(){
        alert("button Click");
    });
</script>
```

이벤트 리스너

이벤트 등록 – 이벤트 등록 메서드 방식

이벤트 함수에 화살표 함수, 혹은 함수 선언문이나 함수 표현식으로 정의한 함수명으로 연결해도 됩니다

함수 표현식으로 정의된 함수는 호이스팅에 의해 선언과 할당이 분리되므로 참조하려는 함수가 addEventListener() 메서드보다 반드시 위에 작성되어야 함

```
const btnEl = document.querySelector("button");
const clickEvent = () => {
    alert("button Click");
}
btnEl.addEventListener("click", clickEvent);
```

이벤트 리스너

이벤트 객체

이벤트가 발생하면 실행되는 함수에는 내부적으로 이벤트 객체가 매개변수로 전달됨

이벤트 함수에 매개변수(event) 지정.

addEventListener() 메서드 내부에 감춰져 있어서 세부 내용을 자세히 알 필요가 없음.

간단하게 이벤트 함수에는 이벤트 객체라는 데이터가 내부적으로 전달되어 호출된다고만 이해

```
<button>클릭</button>
<script>
    const btnEl = document.querySelector("button");
    btnEl.addEventListener("click", function(event){ // 이벤트 객체
        console.log(event);
    })
</script>
```

이벤트 객체

클릭 이벤트의 PointerEvent 객체

속성	설명
clientX	마우스가 클릭된 x좌표(수평 스크롤 포함 X)
clientY	마우스가 클릭된 y좌표(수직 스크롤 포함 X)
pageX	마우스가 클릭된 x좌표(수평 스크롤 포함 O)
pageY	마우스가 클릭된 y좌표(수직 스크롤 포함 O)
screenX	모니터의 왼쪽 위 모서리를 기준으로 마우스가 클릭된 x좌표
screenY	모니터의 왼쪽 위 모서리를 기준으로 마우스가 클릭된 y좌표

이벤트 리스너

이벤트 객체

키보드 이벤트도 자주 사용함

속성	설명
keyCode	키보드에서 눌린 키의 유니코드 값을 반환합니다.
ctrlKey	Ctrl 키가 눌렸으면 true, 그렇지 않으면 false를 반환합니다.
altKey	Alt 키가 눌렸으면 true, 그렇지 않으면 false를 반환합니다.
shiftKey	Shift 키가 눌렸으면 true, 그렇지 않으면 false를 반환합니다.

```
<form>
    <input type="text">
</form>
<script>
    const inputEl = document.querySelector("input");
    inputEl.addEventListener("keydown", function(event){
        console.log(event);
    })
</script>
```

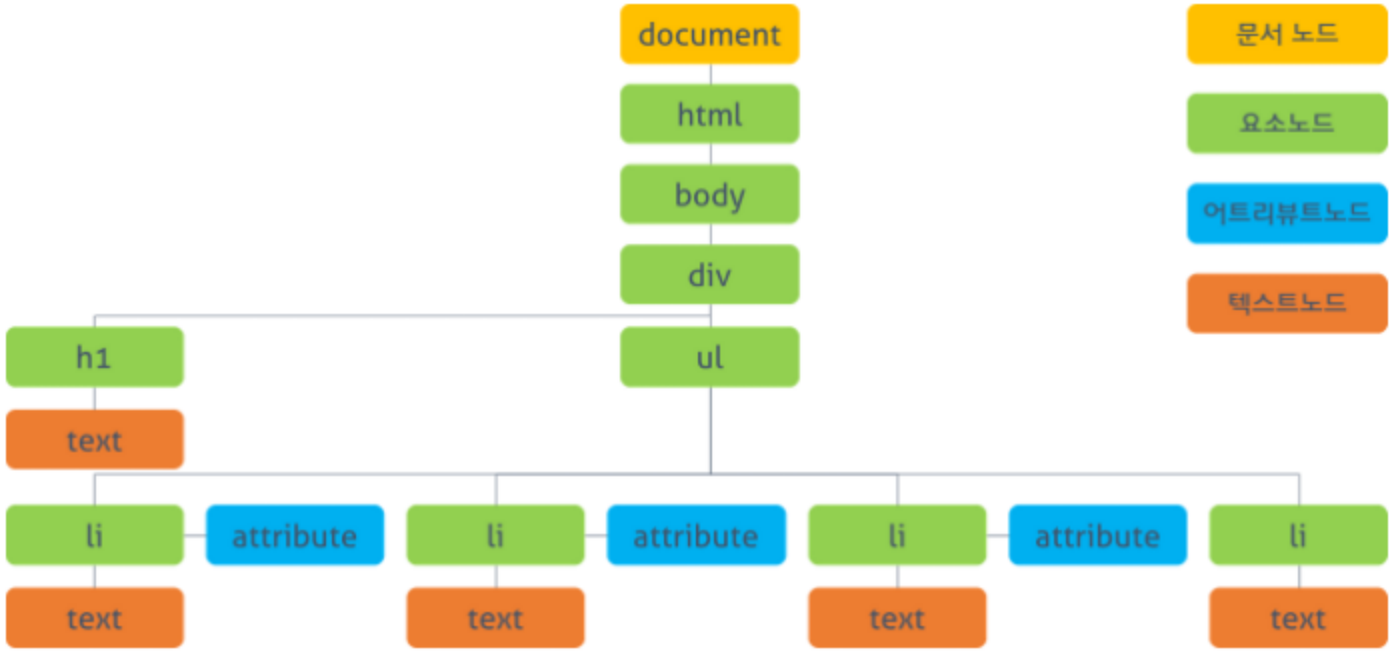
오늘 우리는

DOM

DOM (Document Object Model) 문서객체모델

가장 꼭대기(최상위)에 있는 노드를 루트 노드(root node)

document는 노드가 아니라 객체이므로 여기서는 html을 루트 노드라 부름



콘텐츠 조작하기

속성	설명
textContent	요소 노드의 모든 텍스트에 접근합니다.
innerText	요소 노드의 텍스트 중 웹 브라우저에 표시되는 텍스트에만 접근합니다.
innerHTML	요소 노드의 텍스트 중 HTML 태그를 포함한 텍스트에만 접근합니다.

```
<p id="title">Hello, <span style="display:none;">Javascript!</span></p>
```

```
document.getElementById("title").textContent; // Hello, Javascript!
document.getElementById("title").innerText;   // Hello,
document.getElementById("title").innerHTML;   // Hello, <span style="display: none;">Javascript!</span>
```

노드 추가/삭제

노드 추가

구분	메서드	설명
노드 생성	createElement()	요소 노드를 생성합니다.
	createTextNode()	텍스트 노드를 생성합니다.
	createAttribute()	속성 노드를 생성합니다.
노드 연결	<기준 노드>.appendChild(<자식 노드>)	기준 노드에 자식 노드를 연결합니다.
	<기준 노드>.setAttributeNode(<속성 노드>)	기준 노드에 속성 노드를 연결합니다.

```
<!DOCTYPE html>
<html>
<head>
    <title>Create Node</title>
</head>
<body>
    <script></script>
</body>
</html>
```

이벤트 리스너

주요 이벤트데이터 속성 조작하기

웹 브라우저에서 사용자와의 상호작용으로 발생하는 이벤트는 200여 가지가 넘음

구분	이벤트	설명
마우스 이벤트	onclick	마우스로 클릭하면 발생합니다.
	ondblclick	마우스로 빠르게 두 번 클릭하면 발생합니다.
	onmouseover	마우스 포인터를 올리면 발생합니다.
	onmouseout	마우스 포인터가 빠져나가면 발생합니다.
	onmousemove	마우스 포인터가 움직이면 발생합니다.
	onwheel	마우스 휠(wheel)을 움직이면 발생합니다.
키보드 이벤트	onkeypress	키보드 버튼을 누르고 있는 동안 발생합니다.
	onkeydown	키보드 버튼을 누른 순간 발생합니다.
	onkeyup	키보드 버튼을 눌렀다가 뗀 순간 발생합니다.
포커스 이벤트	onfocus	요소에 포커스가 되면 발생합니다.
	onblur	요소가 포커스를 잃으면 발생합니다.
폼 이벤트	onsubmit	폼이 전송될 때 발생합니다.

감사합니다