08.28_실습 강의자료

1교시

1-01 배경색 바꾸기

▼ 이벤트 리스너와 이벤트 핸들러란?

이벤트 핸들러란 이벤트가 발생했을 때 브라우저에 호출이 되는 함수를 가리킵니다.

이벤트가 발생했을 때 브라우저가 히벤트 핸들러의 호출을 위임하는 것을 이벤트 핸들러 등록이라고 합니다.

이벤트 리스너는 발생하는 이벤트를 감지하는 기능을 한다고 생각할 수 있습니다.

따라서 이벤트 리스너가 등록된 요소에 이벤트가 발생했을때 감지하는 것이 이벤트 리스너이고 이벤트 리스너가 이벤트를 감지하면 실행되는 함수가 이벤트 핸들러라고 할 수 있습니다.

실습을 통해 알아보겠습니다🙂

▼ 이벤트 핸들러를 등록방법은?

이벤트 핸들러를 등록하는 방법은 크게 세가지가 있는데요

1. 이벤트 핸들러 어트리뷰트 방식

   HTML요소의 어트리뷰트 중 onclick과 같이 on접두사와 이벤트의 종류를 나타내는 이벤트 타입으로 이루어진 이벤트 핸들러 어트리뷰트가 있습니다. 이를 이용해서 등록할 수 있죠

```
<div onclick="alert('인라인 방식')">인라인 방식</div>
```

2. 이벤트 핸들러 프로퍼티 방식

   DOM 노드 객체는 이벤트에 대응하는 이벤트 핸들러 프로퍼티를 가지고 있습니다.

   이벤트 핸들러 어르티뷰트와 같은 방식으로 on접두사+이벤트타입으로 이루어져 있죠

이벤트 핸들러 프로퍼티에 함수를 바인딩하면 이벤트 핸들러를 등록할 수 있습니다.

```
...
<div id="property">프로퍼티방식</div>
<script>
        const handlerTwo = document.getElementById("property");
        handlerTwo.onclick = function () {
            alert("두번째 방식");
        };
</script>
...
```

3. addEventListener 메서드 방식

   DOM Level2 에도입된 방식으로 addEventListener를 통해서 이벤트 핸들러를 등록할 수 있습니다.

   `EventTarget.addEventListener(이벤트타입, 이벤트핸들러함수 [, capture 사용여부])`

   로 첫번째 매개변수로 이벤트 타입을 등록해줍니다.

   이때 주의점은 위의1, 2 번 방식과 다르게 on 접두사를 붙이지 않는다는 것입니다.

```
...
<div id="listener">리스너등록</div>
<script>
        const handlerThree = document.querySelector("#listener");
        handlerThree.addEventListener("click", () => {
            alert("세번째 방식");
        });
</script>
...
```

앞의 두가지 방법에는 여러가지 이벤트 핸들러를 등록하기 힘들다는 단점과 수정이 어렵다는 단점이 존재해 이벤트 리스너를 통한 이벤트 등록방법을 가장 권장드립니다.

이들의 가장 큰 차이점은

여러개의 이벤트 핸들러를 등록하는 방법과 이벤트를 제거하는 방법의 차이가 있는데요

코드로 예시를 들겠습니다 🙂

앞의 1, 2 번 방법과는 다르게 addEventListener를 사용하면 동일한 HTML 요소에서 발생하는 동일한 이벤트에 대해 여러개의 이벤트 핸들러를 등록할 수 있습니다.

```
<script>
        const handlerTwo = document.getElementById("property");
        handlerTwo.onclick = function () {
            alert("두번째 방식");
```

```
        };

        // // 불가능
        // handlerTwo.onclick = function () {
        //     console.log("여러개 이벤트 핸들러 등록");
        // };

        const handlerThree = document.querySelector("#listener");
        handlerThree.addEventListener("click", () => {
            alert("세번째 방식");
        });

        // 가능
        handlerThree.addEventListener("click", () => {
            console.log("여러개 이벤트 핸들러 등록");
        });
</script>
```

또한 이벤트 핸들러를 간편하게 제거할 수 있는 장점이 있습니다. 하지만 삭제하기 위해서는 이벤트 핸들러를 등록할때 지켜야 하는 규칙이 있습니다

이벤트를 삭제하기 위해서는 이벤트를 참조할 수 있어야 하므로 이벤트 핸들러함수를 기명함수 또는 외부 함수 참조 형태로 등록해줘야합니다. 실습을 통해 하나씩 확인해 보겠습니다 🙂

이벤트 삭제 예시

https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_element_addeventlistener_remove

tip) 이벤트 한번만 실행되도록하기

한번 실행된 후 이벤트를 제거해주는 방법도 있지만

addEventListener의 3번째 파라미터를 사용하면 쉽게 해결할 수 있습니다.

▼ 자주쓰는 이벤트 종류

마우스 이벤트

이벤트 타입	설명
`mousedown`	마우스 버튼을 누르는 순간
`mouseup`	마우스 버튼을 눌렀다 떼는 순간
`click`	왼쪽 버튼을 클릭한 순간
`dblclick`	왼쪽 버튼을 빠르게 두 번 클릭한 순간

`contextmenu`	오른쪽 버튼을 클릭한 순간
`mousemove`	마우스를 움직이는 순간
`mouseover`	마우스 포인터가 요소 위로 올라온 순간
`mouseout`	마우스 포인터가 요소에서 벗어나는 순간
`mouseenter`	마우스 포인터가 요소 위로 올라온 순간 (버블링이 일어나지 않음)
`mouseleave`	마우스 포인터가 요소에서 벗어나는 순간 (버블링이 일어나지 않음)

키보드 이벤트

이벤트 타입	설명
`keydown`	키보드의 버튼을 누르는 순간
`keypress`	키보드의 버튼을 누르는 순간 ('a', '5' 등 출력이 가능한 키에서만 동작하며, Shift, Esc 등의 키에는 반응하지 않음)
`keyup`	키보드의 버튼을 눌렀다 떼는 순간

포커스 이벤트

이벤트 타입	설명
`focusin`	요소에 포커스가 되는 순간
`focusout`	요소로부터 포커스가 빠져나가는 순간
`focus`	요소에 포커스가 되는 순간 (버블링이 일어나지 않음)
`blur`	요소로부터 포커스가 빠져나가는 순간 (버블링이 일어나지 않음)

입력 이벤트

이벤트 타입	설명
`change`	입력된 값이 바뀌는 순간
`input`	값이 입력되는 순간
`select`	입력 양식의 하나가 선택되는 순간
`submit`	폼을 전송하는 순간

스크롤 이벤트

이벤트 타입	설명
`scroll`	스크롤 바가 움직일 때

윈도우 창 이벤트

이벤트 타입	설명
`resize`	윈도우 사이즈를 움직일 때 발생

▼ html요소에 class를 추가/변경하는 방법

크게 className과 classList를 사용하는 방식이 있습니다.

className은 class 전체를 가져오며 새로운 값을 가져왔을 때 이전의 모든 클래스가 사라지고 덮어쓰기 방식으로 할당됩니다.

classList는 class item들을 하나씩 가져옵니다. 그리고 메서드를 이용해 추가/삭제를 해줍니다.

classList의 메서드

add : 지정한 클래스 값 추가

remove : 지정한 클래스 값 제거

toggle : 지정한 클래스가 있으면 제거, 없으면 추가

className을 잘못 사용하면 이전에 설정한 class들이 모두 사라질 위험이 있기 때문에 초기화가 목적이 아니라면 classList를 사용하시길 권장드립니다.

1-02 화면의 상단으로 이동하기

▼ DOM요소 지정하는 방법은?

DOM요소을 지정하는 방법으로는

Id를 이용할 때 : `getElementById(id)` : 입력된 id를 가진 요소 (단수)

Class를 이용할 때 : `getElementsByClassName(names)` : 입력된 names 클래스를 가진 요소들 (복수)

Tag를 이용할 때 : `getElementsByTagName(name)` : 입력된 name 태그를 가진 요소들(복수)

CSS 선택자를 이용할 때 :

1. querySelector(selectors) : selectors로 선택한 요소중 첫번째 값 (단수)
2. querySelectoraAll(selectors) : selectors로 선택한 모든 요소 (복수)

여러분의 기호에 맞게 사용하시면 됩니다~!

▼ 스크롤을 이동시키는 방법은?

window의 메서드인 scrollTo를 이용해서 원하는 위치로 스크롤을 이동시킬 수 있습니다.

```
window.scrollTo({top, left, behavior})

// top : 세로위치, left: 가로위치, behavior: 스크롤 효과속성
// behavior 옵션
// - auto : 기본값, 바로 위치로 이동
// - smooth : 부드럽게 이동
```

참고 : https://developer.mozilla.org/ko/docs/Web/API/Window/scrollTo

▼ window객체란?

웹 브라우저의 창(window)을 나타내는 객체로 JS의 모든 객체, 전역변수, 전역함수들은 자동으로 window의 프로퍼티가 됩니다. 사실은 우리가 선언한 변수나 함수에도 앞에 window가 붙어있지만 모든 객체를 포함하고 있기때문에 생략해서 사용하는 것입니다.

2교시

1-03 출석하기

▼ querySelector 심화편

1주차에 열심히 배웠던 CSS Selector!! 여기서 한 번 다시 복습해 볼까요?

querySelector를 사용하면 우리가 배웠던 CSS 선택자를 사용해서 DOM요소를 택할 수 있습니다!!

▼ HTML안에 text 넣기!

html 안에 text를 쓰는방식 중 innerText와 innerHTML, textContext의 차이를 알아보겠습니다.

element.innerText()

element안의 text 값으로 넣어줍니다. 사용자가 읽을 수 있는 text들을 가져오거나 설정할 수있습니다.

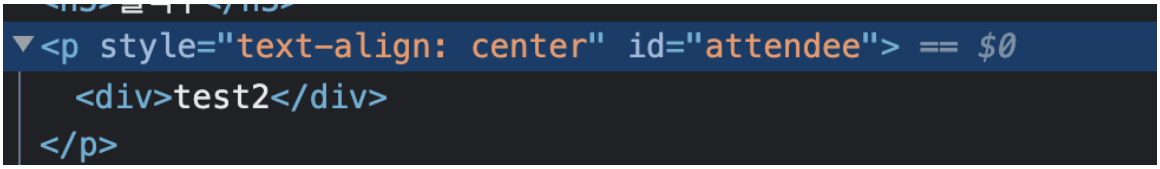
(display : none 으로 작성된 코드는 볼 수 없죠 😞)

```
<p style="text-align: center" id="attendee"><div>test1</div>
</p> == $0
```

참고 : https://developer.mozilla.org/ko/docs/Web/API/HTMLElement/innerText

element.innerHTML()

innerText와는 달리 HTML 코드로 인식하고 핸들링합니다.



참고 : https://developer.mozilla.org/ko/docs/Web/API/Element/innerHTML

node.textContent()

태그의 자손의 텍스트 콘텐츠들을 가져옵니다.

또한 innerText()와는 다르게 script, style 요소를 포함한 모든 요소를 가져오며

숨겨진 요소도 모두 반환합니다.

textContent는 위의 두가지 보다 성능적으로, 보안적으로 뛰어납니다.

참고 : https://developer.mozilla.org/ko/docs/Web/API/Node/textContent

1-04 버튼 활성화시키기

▼ window.alert() 와 alert()의 차이점은?

window는 브라우저의 전역 객체입니다. 사실 브라우저에서 사용되는 모든 메서드 들이 window. 으로 시작하는 것이죠

모든 곳에 window. 가 붙으니 생략 가능하게 하자~ 라고 약속을 했기 때문에

window.는 생략이 가능합니다 🙂

cf. alert가 못생겼죠! 간편하게 alert를 꾸미는 방법은 sweetalert2라는 라이브러리를 이용해보세요~

https://sweetalert2.github.io/#download

▼ 유저가 pw부터 친다면..?

현재 코드에서는 pw에만 이벤트 리스너를 등록해놨기 때문에 pw부터 치고 id를 친다면 원하는대로 동작하지 않습니다.

유저가 뭐를 먼저 입력하든 동일하게 작동하기 위해서는 어떻게 코드를 작성해야 할까요?

아래와 같이 코드를 수정해 줄 수 있습니다.

```
function activateBtn() {
    /* 여기에 작성해주세요 */
    if (id.value.length && pw.value.length) {
        loginButton.classList.remove("deactivatedColor");
        loginButton.classList.add("activatedColor");
    } else {
        loginButton.classList.add("deactivatedColor");
        loginButton.classList.remove("activatedColor");
    }
}

id.addEventListener("keyup", activateBtn);
pw.addEventListener("keyup", activateBtn);
```

1-05 N번째 요소 만들기

▼ document.write()의 문제점은?

이전에 배웠던 document.write는 가장 간편하게 js를 이용해서 html을 작성할 수 있지만 여러가지 문제점이 있습니다. 내가 원하는 곳에 html 코드를 작성하기 어렵다는 점입니다.

document.write는 호출된 시점에서만 작성됩니다. 따라서 이번 문제처럼 ul 안에서 사용하기 불편합니다.

물론 가능은 합니다.

```
<body>
    <ul>
        <script src="index.js"></script>
    </ul>
</body>
```

이런식으로 작성해주면 가능하지만 코드만 봐도 다른 단점들이 막 생길 것 같은게 느껴지시죠??

그러면 어떻게 해야 할까요?

▼ JS로 HTML에 태그를 추가하는 방법은?

우리가 1주차에 배웠던 HTML을 생각해보면

HTML을 추가하려면 무엇이 필요할까요?

먼저 어떤 태그를 사용할 것인지, 그리고 태그에 어떤 속성을 넣을 것인지, 어떤 내용을 담을 것인지로 html태그를 만들어 줄 수 있겠죠? 그리고 어디에 추가할 것인지도 정해줘야겠죠?

첫번째로 어떤 태그를 사용할 것인지는 **createElement( )** 라는 메서드를 사용해서 원하는 태그를 생성할 수 있습니다.

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

참고 : https://developer.mozilla.org/ko/docs/Web/API/Document/createElement

추가 해줄 속성으로는 **setAttribute( )** 라는 메서드를 사용할 수 있는데요

```
addTag.createElement('li').setAttribute("속성" : "값")
```

위의 방법으로 속성들을 추가해줄 수 있습니다.

(class를 추가하는 다른 방법은 1번 문제에서 배웠었죠?)

어떤 내용을 담을지는 여러가지 방법이 있지만 이번 실습에서는 2가지 방법을배워보겠습니다.

먼저 직접 text를 넣어주는 것입니다. 3번 문제에서 배웠던 text를 넣는 방법을 사용할 수도 있습니다.

```
const addTag = document.createElement("li");
    addTag.innerText = `${i}번째 문장`;
    addTag.innerHTML = `${i}번째 문장`;
    addTag.textContent = `${i}번째 문장`;
    myUl.appendChild(addTag);
```

다음으로는 지시사항에 나와있는 것처럼 createTextNode()를 사용하는 방법입니다.

```
const addTag2 = document.createElement("li");
const text = document.createTextNode(`${i}번째 문장`);
addTag2.appendChild(text);
```

이렇게 createTextNode로 text node를 생성 후 이 텍스트 노드를 우리가 만든 element에 추가하는 방법입니다.

마지막으로 어느 위치에 넣어줄지를 정해야하는데 appendChild**( )**를 사용할 수 있습니다.

appendChild()는 지정해놓은 태그의 자식태그중 가장 마지막에 새로 생성한 태그를 넣어줍니다.

```
myUl.appendChild(addTag);
```

참고 : https://developer.mozilla.org/ko/docs/Web/API/Node/appendChild

3교시

1-06 태그의 속성에 따라 하이라이트하기

▼ JS로 CSS 속성 바꾸는 방법은?

DOM요소의 style을 바꾸는 방법은

1. style을 인라인 방식으로 넣는 방법
2. 내부 스타일시트 추가/변경 방식
3. 외부 스타일시트을 추가/변경 방식

먼저 인라인 방식은 DOM요소.style 뒤에 넣고 싶은 CSS값을 카멜케이스 방식을 이용해서 넣어주면 됩니다.

```
str.style.color = "blue"
```

내부 스타일 시트를 통해 넣는 방식은 먼저 `document.createElement('style');`을 통해 내부 스타일 시트를 만들어주고 그 안에 style값을 넣어주는 방식입니다.

```
let styleTest = document.createElement('style');
styleTest.innerHTML="div{height:30px;} a{color:#fff;}";
document.head.appendChild(styleTest);
```

외부 스타일 시트를 통해 넣는 방식은 CSS파일에 따로 style을 작성한 후 HTML에 class를 추가해서 넣는 방식입니다. 1번문제에서 배웠던 `classList.add`를 통해 작성한 css style을 추가해주는방식입니다.

```
boxs.classList.add("on"); // DOM요소인 boxs에 on이라는 className을 추가함
```

추가+

▼ HTMLCollection와 NodeList와 Array 차이점

getElementsByTagName와 같은 getElementsBy… 으로 요소를 선택했을 때는 HTMLColelction으로

querySelectorAll으로 요소를선택했을 때는 NodeList로

가져오는것을 보실 수 있습니다.

```
getElementsByTagName:  ▶HTMLCollection(3) [strong, strong, strong]
querySelectorAll:  ▶NodeList(3) [strong, strong, strong]
```

HTMLCollection과 NodeList는 모두 유사 배열입니다.

(배열인지 판단하는 메서드 Array.isArray():
https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array/isArray)

저번주에 배웠던 ES6문법 중 forEach, map, filter, reduce 등은 모두 '배열'의 함수로 유사배열일 HTMLCollection과 NodeList에서는 사용할 수 없습니다.

그러면 유사배열은 어떻게 배열 처럼 쓸 수 있을까요?

▼ 유사배열을 배열로 바꾸는 방법은?

유사배열을 배열로 바꾸는 4가지 방법을 알려드리겠습니다.

Array.from 사용

```
// 1. Array.from
    let test1 = Array.from(words);
    console.log("test1 ", test1);
    test1.map((item) => (item.style.color = "blue"));
```

slice.call 사용

```
// 2. Array.prototype.slice.call
    let test2 = [].slice.call(words);
```

```
    console.log("test2", test2);
    test2.map((item) => (item.style.color = "blue"));
```

map.call 사용

```
// 3. Array.prototype.map.call
    let test3 = [].map.call(words, (item) => (item.style.color = "blue"));
    console.log("test", test3);
```

구조분해할당 사용

```
// 4. 구조분해할당 : [...유사배열]
    let test4 = [...words];
    console.log("test4", test4);
    test4.map((item) => (item.style.color = "blue"));
```