HTML 태그 접근

getElement

- **getElement**

  HTML 요소의 Class, Name, Id, Tag 값을 이용하여 요소에 접근

- **querySelector**

  CSS 선택자를 사용하여 요소에 접근

- **getElement vs querySelector**

  일반적으로 getElement가 더 많은 브라우저에서 지원되며 *속도가 빠르기 때문에 더 많이 사용되나, 성능보다는 개발의 편의성을 위해 querySelector를 사용하는 개발자도 있음

  * HTML Collection vs NodeList

getElement의 종류

- **getElement의 종류**

id는 동일한 값을 가질 수 없기 때문에 getElement
Tag, Class, Name은 동일한 값을 여러 태그에서 가질 수 있기 때문에 getElements

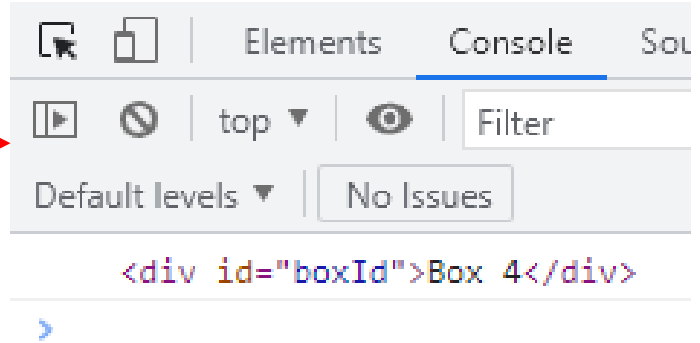
종류	설명	반환 타입
getElementById()	id 값을 가진 요소를 찾아 반환	객체
getElementsByTagName()	지정된 태그 이름을 가진 요소를 찾아 반환	HTMLCollection
getElementsByClassName()	지정된 클래스 이름을 가진 요소를 찾아 반환	HTMLCollection
getElementsByName()	지정된 name 값을 가진 요소를 찾아 반환	NodeList

getElementByID() 예시

- **getElementById() 사용방법**

  const 변수명 = document.getElementById("id값");

```
<div>Box 1</div>
<div>Box 2</div>
<div>Box 3</div>
<div id="boxId">Box 4</div>
<script>
    const boxesId = document.getElementById("boxId");
    console.log(boxesId);
</script>
```

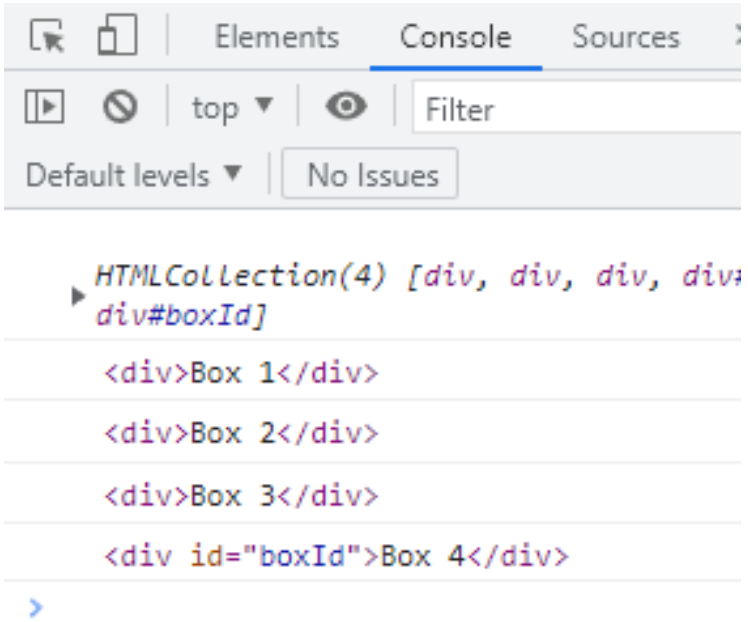


getElementsByTagName() 예시

- **getElementsByTagName() 사용방법**

  const 변수명 = document.getElementsByTagName("tag값");

```
<div>Box 1</div>
<div>Box 2</div>
<div>Box 3</div>
<div id="boxId">Box 4</div>

<script>
    const boxesTag = document.getElementsByTagName("div");
    console.log(boxesTag); // HTMLCollection(4) [div, ...

    for(let i=0; i<boxesTag.length; i++) {
        console.log(boxesTag[i]);
    }
</script>
```

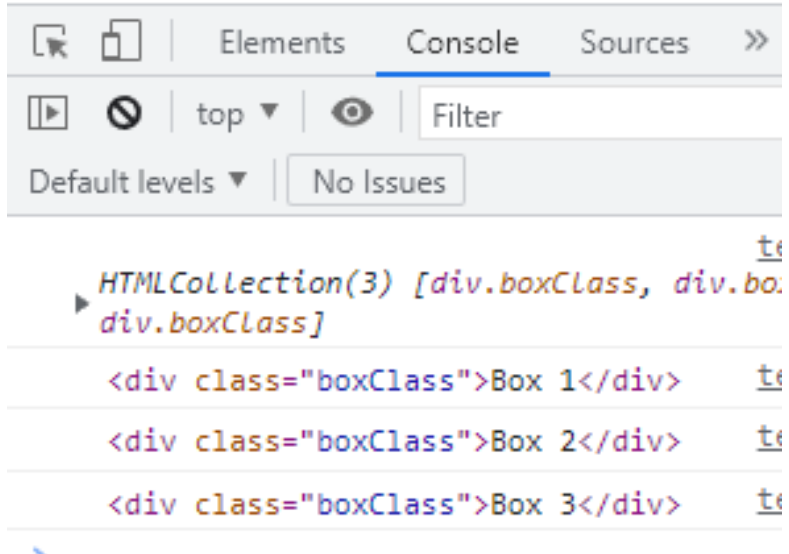


getElementsByClassName() 예시

- **getElementsByClassName() 사용방법**

  const 변수명 = document.getElementsByClassName("class값");

```
<div class="boxClass">Box 1</div>
<div class="boxClass">Box 2</div>
<div class="boxClass">Box 3</div>
<div id="boxId">Box 4</div>

<script>
    const boxesClass = document.getElementsByClassName("boxClass");
    console.log(boxesClass); // HTMLCollection(3) [div.boxClass, ...

    for(let i=0; i<boxesClass.length; i++) {
        console.log(boxesClass[i]);
    }
</script>
```

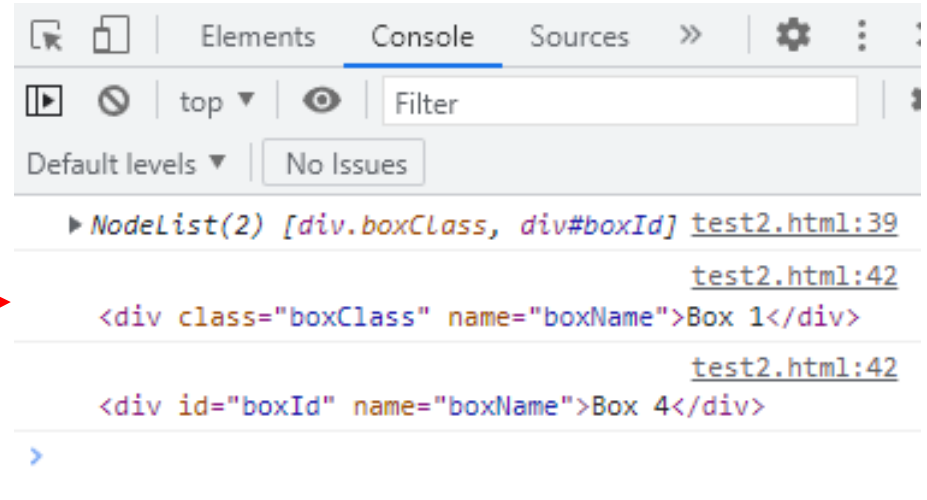


getElementsByName() 예시

- **getElementsByName() 사용방법**

const 변수명 = document.getElementsByName("name값");

```
<div class="boxClass" name="boxName">Box 1</div>
<div class="boxClass">Box 2</div>
<div class="boxClass">Box 3</div>
<div id="boxId" name="boxName">Box 4</div>

<script>
    const boxesName = document.getElementsByName("boxName");
    console.log(boxesName); // NodeList(2) [div.boxClass, ...

    for(let i=0; i<boxesName.length; i++) {
        console.log(boxesName[i]);
    }
</script>
```



데이터 출력

데이터 출력 종류

- **console.log(“내용”);**

  브라우저의 개발자 도구 → 콘솔 창에서 내용을 출력,
  개발자 도구 내부적으로 toString() 메서드를 사용하여 변수의 내용을 출력할 수 있음

- **document.write(“내용“);**

  웹 페이지 화면에서 내용을 출력 → 변수의 내용 출력 불가, 별도 프로퍼티 사용 필요

- **alert(“내용”);**

  팝업창으로 내용을 출력 → 변수의 내용 출력 불가, 별도 프로퍼티 사용 필요

- **innerHTML = “내용”;**

  해당 태그 요소의 내용을 변경하여 출력

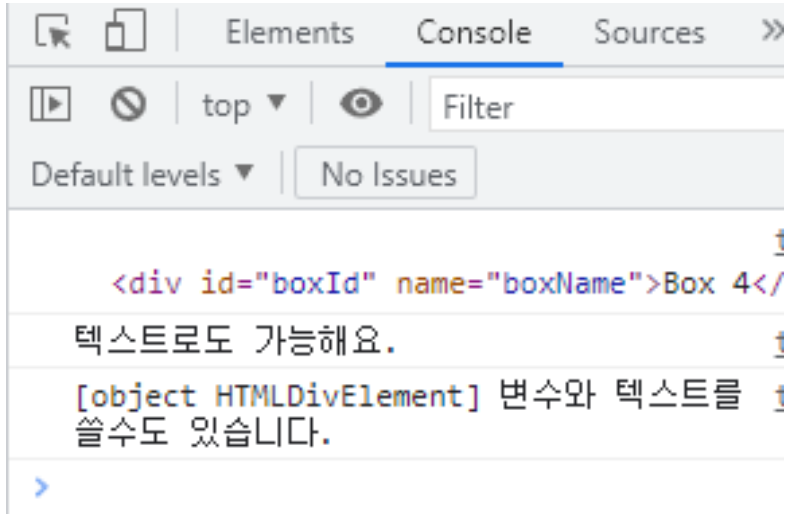
console.log("내용"); 예시

- **[object HTMLDivElement]**

boxesId 변수는 객체이므로 문자열과 연결할 수 없기 때문에, textContent 또는 outerHTML 프로퍼티를 사용하여 문자열과 연결해야 함

ex)
console.log(boxesId.textContent + " 변수와 문자열을 쓸 수도 있습니다.");
console.log(boxesId.outer + " 변수와 문자열을 쓸 수도 있습니다.");

```
<div class="boxClass" name="boxName">Box 1</div>
<div class="boxClass">Box 2</div>
<div class="boxClass">Box 3</div>
<div id="boxId" name="boxName">Box 4</div>

<script>
    const boxesId = document.getElementById("boxId");
    console.log(boxesId);
    console.log("텍스트로도 가능해요.");
    console.log(boxesId+" 변수와 텍스트를 쓸 수도 있습니다.");
</script>
```

document.write("내용"); 예시

- **[object HTMLDivElement]**

console.log()의 경우 개발자 도구에서 내부적으로 toString() 메서드를 사용하기 때문에 변수의 내용이 출력이 가능 하지만, document.write()는 객체를 문자열로 변환하여 출력하기 때문에 값이 다름.

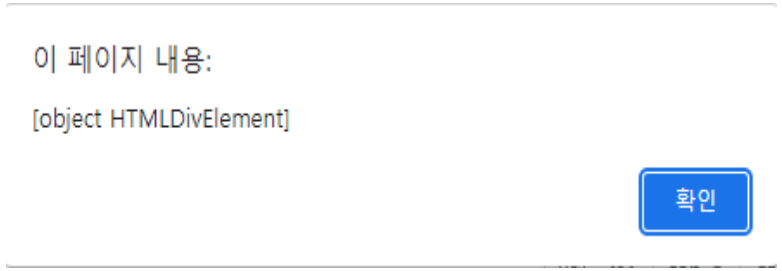
console.log()와 마찬가지로 textContent 또는 outerHTML을 사용하여 해결할 수 있다.

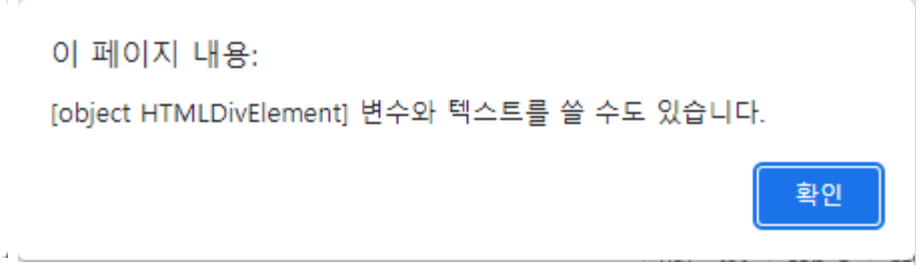
```
<div class="boxClass" name="boxName">Box 1</div>
<div class="boxClass">Box 2</div>
<div class="boxClass">Box 3</div>
<div id="boxId" name="boxName">Box 4</div>

<script>
    const boxesId = document.getElementById("boxId");
    document.write(boxesId);
    document.write("텍스트로도 가능해요.");
    document.write(boxesId+" 변수와 텍스트를 쓸 수도 있습니다.");
</script>
```

Box 1
Box 2
Box 3
Box 4
[object HTMLDivElement]텍스트로도 가능해요.[object HTMLDivElement] 변수와 텍스트를 쓸 수도 있습니다.

alert("내용"); 예시

```
<div class="boxClass" name="boxName">Box 1</div>
<div class="boxClass">Box 2</div>
<div class="boxClass">Box 3</div>
<div id="boxId" name="boxName">Box 4</div>

<script>
    const boxesId = document.getElementById("boxId");
    alert(boxesId);
    alert("텍스트로도 가능해요.");
    alert(boxesId+" 변수와 텍스트를 쓸 수도 있습니다.");
</script>
```





innerHTML = “내용”; 예시

- 설명

위의 데이터 출력 함수들과는 다르게, HTML 요소의 내용을 변경하는 프로퍼티이다.

```
<div class="boxClass" name="boxName">Box 1</div>
<div class="boxClass">Box 2</div>
<div class="boxClass">Box 3</div>
<div id="boxId" name="boxName">Box 4</div>

<script>
    const boxesId = document.getElementById("boxId");
    boxesId.innerHTML = "Box 4의 내용을 변경";
</script>
```


Box 1
Box 2
Box 3
Box 4의 내용을 변경

데이터 입력

데이터 입력 종류

- **confirm(“질문 내용”);**

  질문에 대해 예/아니오의 결과를 얻고 싶을 때 사용

- **prompt(“질문 내용”);**

  사용자에게 어떠한 텍스트를 전달받고 싶을 때 사용

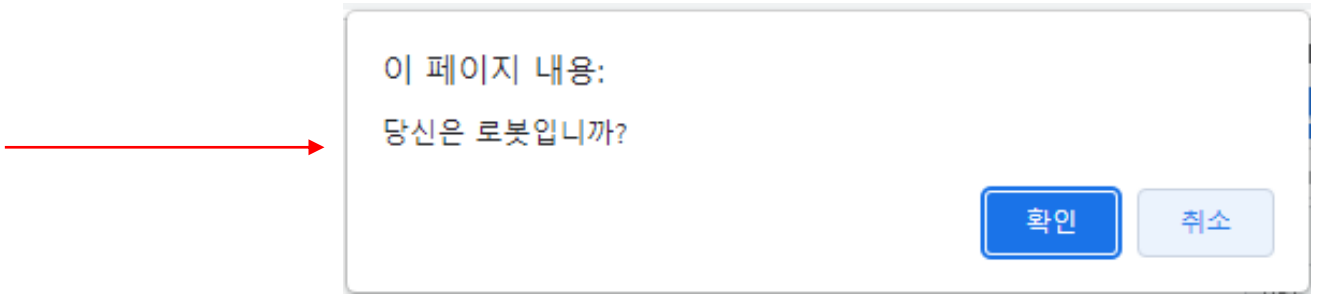
confirm();

- **설명**

질문에 대한 답(예/아니오)을 변수에 저장하고, 변수의 값을 통해 사용자가 어떠한 답을 했는지 리턴값으로 확인할 수 있음

*리턴값 : 확인(true), 취소(false)

```
<div class="boxClass" name="boxName">Box 1</div>
<div class="boxClass">Box 2</div>
<div class="boxClass">Box 3</div>
<div id="boxId" name="boxName">Box 4</div>

<script>
    const question = confirm("당신은 로봇입니까?");
    console.log(question);
</script>
```



prompt();

- 설명

단순 예/아니오의 대답이 아닌, 사용자에게 어떠한 텍스트를 전달받을 때 사용되며 확인을 누르면 데이터 필드에 입력된 값이 리턴 되고 취소를 누르면 null 값이 리턴 됨

```
<div class="boxClass" name="boxName">Box 1</div>
<div class="boxClass">Box 2</div>
<div class="boxClass">Box 3</div>
<div id="boxId" name="boxName">Box 4</div>

<script>
    const question = prompt("당신의 이름은 무엇입니까?");
    console.log(question);
</script>
```