

## 05. DOM의 기초

---

# DOM과 DOM 트리

---

# 문서 객체 모델(DOM)이란

- 자바스크립트를 이용하여 웹 문서에 접근하고 제어할 수 있도록 객체를 사용해 웹 문서를 체계적으로 정리하는 방법
- 웹 문서를 구조화한 DOM 트리(DOM tree)와 이벤트 등을 정리해 놓은 표준

웹에서 자바스크립트를 사용하는 이유는 어떤 조건이 주어지거나 사용자의 동작이 있을 때 웹 문서 전체 또는 일부분이 동적으로 반응하게 하는 것 → 이렇게 하려면 웹 문서의 모든 요소를 따로 제어할 수 있어야 한다.

(예) 웹 문서에 텍스트와 이미지가 들어 있다면

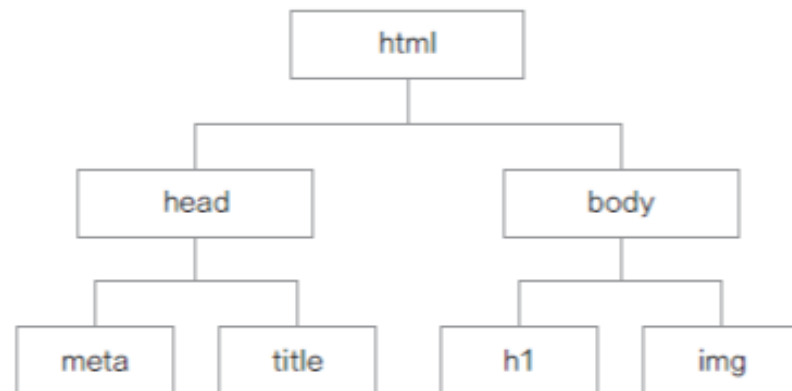
- 웹 브라우저는 마크업 정보를 보면서 텍스트 단락이 몇 개이고 그 내용이 무엇인지 살펴본다
- 이미지가 몇 개이고 이미지 파일 경로는 어떠한지 대체 텍스트는 무엇인지도 파악해서 이미지별로 정리해서 인식한다.
- 텍스트와 이미지 요소를 브라우저가 제어하려면 두 요소를 따로 구별해서 인식해야 한다.
- 마크업을 보면서 요소 사이의 포함 관계도 알아야 한다.

# DOM 트리

- 웹 문서에 있는 요소들 간의 부모, 자식 관계를 계층 구조로 표시한 것
- 나무 형태가 되기 때문에 "DOM 트리"라고 함.
- 노드(node) : DOM 트리에서 가지가 갈라져 나간 항목
- 루트 노드(root node) : DOM 트리의 시작 부분(html)

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>DOM Tree 알아보기</title>
</head>
<body>
  <h1>Do it!</h1>
  
</body>
</html>
```

html 요소는 head, body의 부모 요소

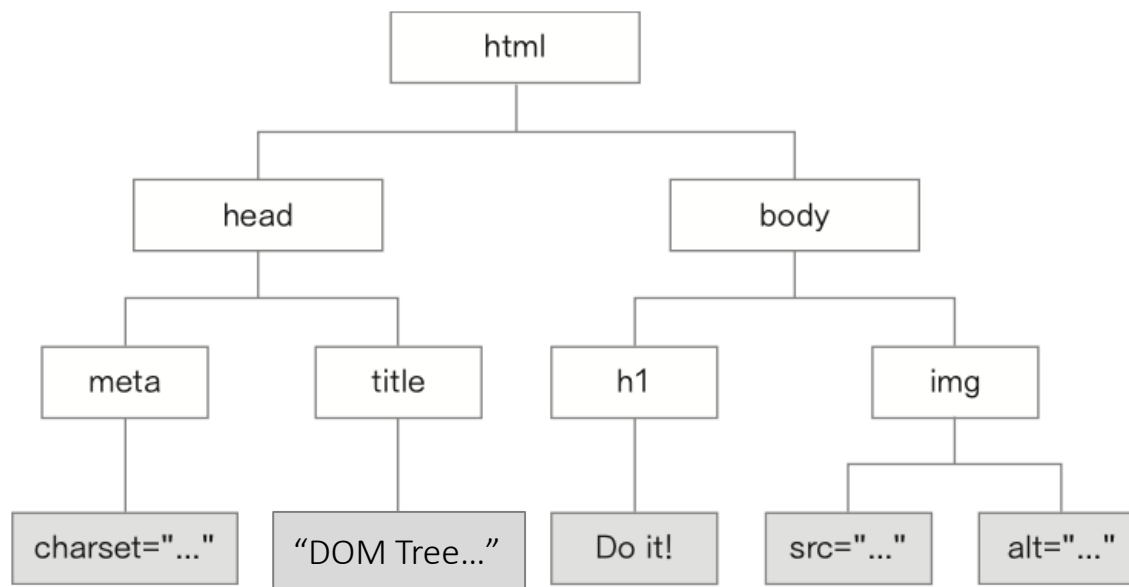


h1 요소는 body의 자식 요소

# DOM 트리

- DOM은 문서의 요소 뿐만 아니라 각 요소의 내용과 속성도 자식으로 나타냄

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>DOM Tree 알아보기</title>
</head>
<body>
  <h1>Do it!</h1>
  
</body>
</html>
```



---

# 웹 요소에 접근하기

---

# 웹 요소에 접근하기

웹 문서에서 원하는 요소를 찾아가는 것을 “접근한다(access)”고 함  
(예) 제목의 글자색을 바꾸고 싶다면 우선 제목까지 ‘접근해야’ 함.

## querySelector(), querySelectorAll() 함수

querySelector(선택자)

querySelectorAll(선택자)

### 선택자

id 이름 앞에는 해시 기호(#), class 이름 앞에는 마침표(.), 태그는 기호 없이 태그명 사용

### 반환 값

- querySelector( ) 메서드는 한 개의 값만 반환
- querySelectorAll( ) 메서드는 반환 값이 여러 개일 때 모두 반환 → 노드 리스트로 저장됨

05Windex.html을 웹 브라우저 창에 불러온 후 콘솔 창을 연다.

## 웹 요소 하나만 접근할 때

```
document.querySelector("h1")
```

```
document.querySelector("#profile")
```

```
> document.querySelector("h1")
< <h1>My Profile</h1>
> document.querySelector("#profile")
< ▼ <div id="profile"> flex
    
    ▶ <div id="desc">...</div>
    </div>
> |
```

## 웹 요소 여러 개에 접근할 때

```
document.querySelectorAll(".user")
```

```
> document.querySelectorAll(".user")
< ▼ NodeList(3) [p.user, p.user, p.user] ⓘ
    ▶ 0: p.user
    ▶ 1: p.user
    ▶ 2: p.user
    length: 3
    ▶ [[Prototype]]: NodeList
>
```



# getElement~ 함수

- 예전부터 사용하던 방법
- id나 class, 태그명을 사용해서 접근

## getElementById( )

```
document.getElementById("id명")
```

## getElementsByClassName( )

```
document.getElementsByClassName("클래스명")
```

## getElementsByTagName( )

```
document.getElementsByTagName("태그명")
```

## getElement\*() 와 querySelector()의 차이

- id값이나 class값, 태그 이름을 사용해서 접근하는 것은 같다
- querySelector()를 사용하면 둘 이상의 선택자를 조합해서 접근할 수 있음 (예, #detail > p)

# 웹 요소 내용 가져오기 및 수정하기

접근한 요소의 텍스트 내용을 가져오거나 지정할 때는 innerText, innerHTML, textContent 프로퍼티 사용

- innerText : 순수 텍스트를 가져오거나, 해당 요소에 텍스트 지정
- innerHTML : 태그와 함께 텍스트를 가져오거나, 해당 요소에 태그와 함께 텍스트 지정
- textContent : 텍스트를 가져오되, 화면에 보이는데로가 아니라 소스에 있는대로 가져옴  
(화면에서 감춘 요소에서도 내용을 가져올 수 있고, 소스에 공백이 여러 개일 경우 그 공백도 모두 가져옴)

```
요소.innerText
```

```
요소.innerHTML
```

```
요소.textContent
```

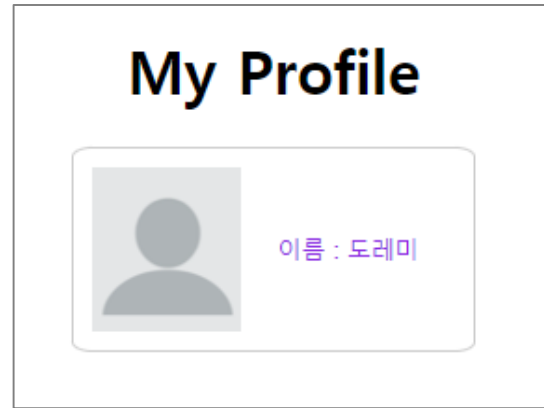
```
요소.innerText = "내용"
```

```
요소.innerHTML = "내용"
```

```
요소.textContent = "내용"
```

05Wjs-content-1.html

```
<div id="desc">
  <p class="user">이름 : 도레미</p>
  <p class="user">주소 : somewhere</p>
  <p class="user">연락처 : 1234-
5678</p>
</div>
```



```
document.querySelector("#desc").inne
rText
```

```
> document.querySelector("#desc").innerText
< '이름 : 도레미'
>
```

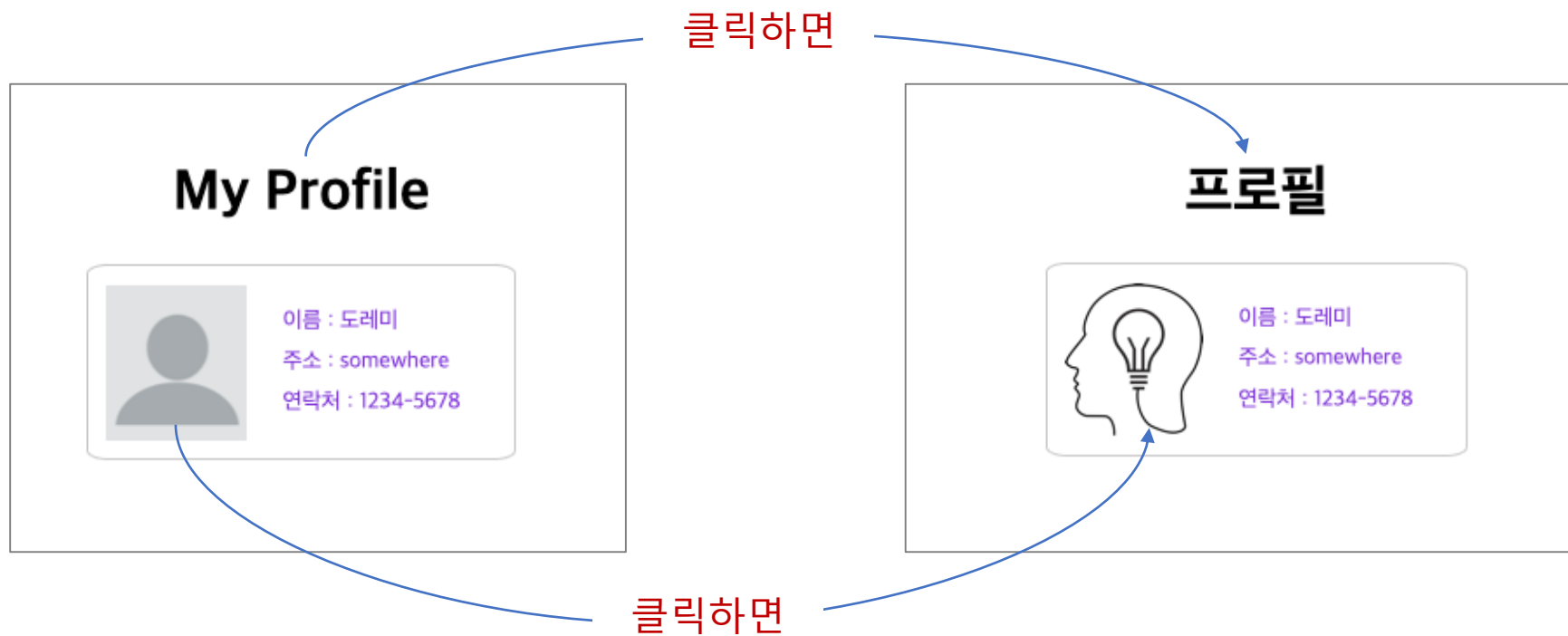
```
document.querySelector("#desc").inne
rHTML
```

```
> document.querySelector("#desc").innerHTML
< '\n      <p class="user">이름      :      도레미</p>\n      <p class="us
er" style="display:none">주소      :      somewhere</p>\n      <p class
="user" style="display:none">연락처      :      1234-5678</p>\n      '
> |
```

```
document.querySelector("#desc").text
Content
```

```
> document.querySelector("#desc").textContent
< '\n      이름      :      도레미\n      주소      :      somewhere\n
연락처      :      1234-5678\n      '
> |
```

(예) 클릭해서 텍스트 내용 또는 이미지 바꾸기

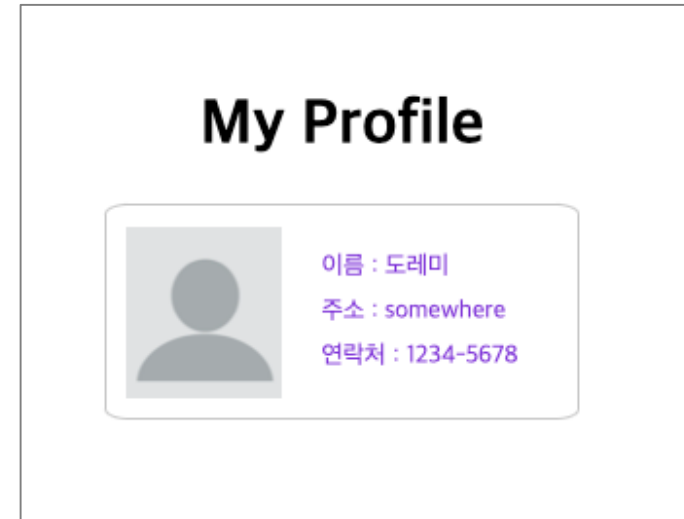


## (예) 클릭해서 텍스트 내용 또는 이미지 바꾸기

### 1) 문서 소스 확인하기

05Wjs-content-2.html

```
<h1 id="title">My Profile</h1>
  <div id="profile">
    
    <div id="desc">
      <p class="user">이름 : 도레미</p>
      <p class="user">주소 :
somewhere</p>
      <p class="user">연락처 : 1234-
5678</p>
    </div>
  </div>
```



## (예) 클릭해서 텍스트 내용 또는 이미지 바꾸기

### 2) 제목 부분 클릭했을 때 변경하기

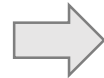
click 이벤트가 발생했을 때, 요소에 직접 함수를 연결해서 사용

변수.onclick = 함수

05WjsWjs-content-2.js

```
const title = document.querySelector("#title");

title.onclick = function() {
  title.innerText = "프로필";
}
```



화살표 함수를 사용하면

```
const title = document.querySelector("#title");

title.onclick = () => title.innerText = "프로필";
```

## (예) 클릭해서 텍스트 내용 또는 이미지 바꾸기

### 3) 이미지 부분 클릭했을 때 변경하기

이미지를 변경하려면 이미지에 접근한 후 src 속성 값을 바꿔준다

이미지 요소.src = 이미지 파일 경로

05WjsWjs-content-2.js

```
const title = document.querySelector("#title");  
const pflImage = document.querySelector("#profile img");  
  
title.onclick = () => title.innerText = "프로필";  
pflImage.onclick = () => pflImage.src = "images/pf2.png";
```



---

# 자바스크립트로 스타일 수정하기

---



# CSS 속성에 접근하기


자바스크립트를 이용하면 스타일 속성의 값을 가져오거나 원하는 값으로 수정할 수 있습니다.

→ 웹 문서에서 다양한 효과를 만들 수 있습니다.

요소.style.속성명

예) 글자색 수정하려면 style.color

배경색 수정하려면 style.backgroundColor

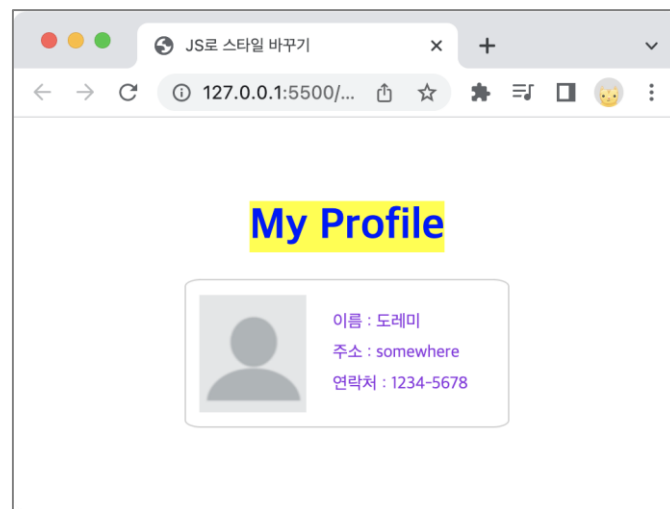
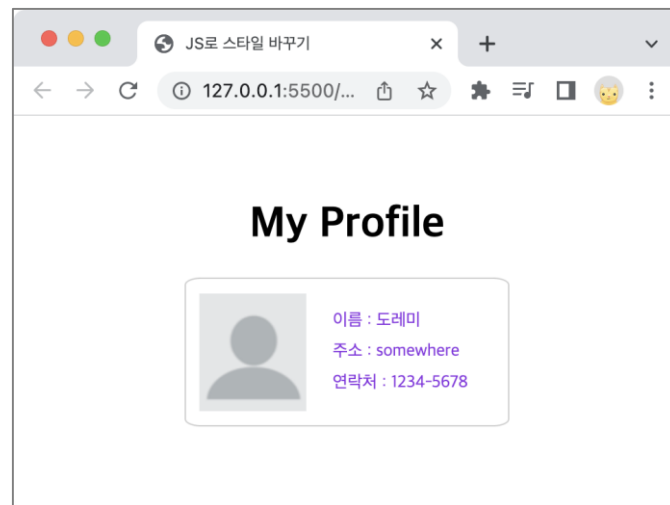


background-color, font-size 처럼 하이픈을 사용해 두 단어를 연결한 CSS 속성  
→ backgroundColor, fontSize 처럼 하이픈 없이 사용. 둘째 단어는 대문자로

(예) 제목 부분을 클릭했을 때 제목의 글자색과 글자 배경색 바꾸기

05Wjs-css.html, 05WjsWjs-css.js

```
const title = document.querySelector("#title");
title.onclick = () => {
    title.style.backgroundColor = "yellow";
    title.style.color = "blue";
}
```



# classList 프로퍼티

- 두 개 이상의 class 스타일이 적용되었을 경우 class 스타일 정보를 담아두는 프로퍼티.
- `classList`를 사용해서 적용 중인 class 스타일을 제거할 수도 있고 새로운 class 스타일을 추가할 수도 있다

05Wclasslist-0.html

```
<div id="desc">
  <p class="user clicked">이름 : 도레미
</p>
  <p class="user">주소 : somewhere</p>
  <p class="user">연락처 : 1234-5678</p>
</div>
```

콘솔 창에서 확인하기

```
document.querySelector("#desc p").classList
```

```
> document.querySelector("#desc p").classList
< ▶ DOMTokenList(2) ['user', 'clicked', value: 'user clicked']
>
```

```
> document.querySelector("#desc p").classList
< ▼ DOMTokenList(2) ['user', 'clicked', value: 'user clicked']
  0: "user"
  1: "clicked"
  length: 2
  value: "user clicked"
  ▶ [[Prototype]]: DOMTokenList
>
```

# 클래스 스타일 추가하기 및 삭제하기

새로운 클래스 스타일을 추가하거나 (이 경우, 클래스 스타일이 만들어져 있어야 함)  
기존에 적용 중인 클래스 스타일을 제거할 수 있습니다.

```
요소.classList.add(클래스명)
```

```
요소.classList.remove(클래스명)
```

# add() – 클래스 스타일 추가

05WcssWstyle.css

```
h1 {  
  font-size:2rem;  
  margin-bottom:20px ;  
}  
  
.clicked {  
  background-color:yellow;  
}
```

05Wclasslist-1.html, 05WjsWclasslist-1.js

```
const title = document.querySelector("#title");  
  
title.onclick = () => {  
  title.classList.add("clicked");  
}
```

제목을 클릭했을 때 .clicked 스타일이 추가됨

# contains() – 특정 클래스 스타일 있는지

앞의 예제에서 제목을 클릭하면 배경색과 글자색이 바뀌는데, 원래 상태로 되돌아가지는 않는다.

추가했던 클래스 스타일을 제거하면 원래대로 돌아간다

그러기 위해서는 특정 클래스 스타일이 있는지 체크할 수 있어야 한다

→ `classList`의 `contains()` 함수 사용

```
요소.classList.contains(클래스명)
```

# contains() – 특정 클래스 스타일 있는지

- 1) contains() 함수를 사용해서 .clicked 스타일이 있는지 살펴보고,
- 2) .clicked 스타일이 있다면 remove()를 사용해서 제거한다

```
const title = document.querySelector("#title");
```

```
title.onclick = () => {
```

```
    if(!title.classList.contains("clicked"))
```

.clicked가 없으면 추가

```
){
```

```
    title.classList.add("clicked");
```

.clicked가 있으면 삭제

```
    } else {
```

```
        title.classList.remove("clicked");
```

```
    }
```

```
}
```

# toggle() – 특정 스타일 토글

특정 클래스를 추가하거나 삭제하기를 반복할 경우에는 `classList`의 `toggle()` 함수가 더 편리하다.

`요소.classList.toggle(클래스명)`

```
const title = document.querySelector("#title");

title.onclick = () => {
  if(!title.classList.contains("clicked"))
){
  title.classList.add("clicked");
} else {
  title.classList.remove("clicked");
}
```



```
const title = document.querySelector("#title");

title.onclick = () => {
  title.classList.toggle("clicked");
}
```



---

# DOM에서 폼 다루기

---

# 폼 요소에 접근하기

19Worder.html을 웹 브라우저에서 열고 콘솔 창에서 연습합니다

상품 정보

상품 :

갯수 :

주문 정보

이름 :

연락처 :

주소 :

주문하기

```
<fieldset>
  <legend>주문 정보</legend>
  <ul>
    <li>
      <label class="field" for="order-name">이름 :
      </label>
      <input type="text" class="input-box" id="or
der-name" name="order-name">
    </li>
    .....
  </ul>
</fieldset>
```

## id를 사용해 접근하기

```
document.querySelector("#order-name")
```



## 텍스트 필드에 입력한 내용 가져오기

```
요소.value
```

```
document.querySelector("#order-name")
).value
```



## name 속성 값을 사용해 접근하기

혹시 다른 사람이 작성해 놓은 폼 소스에 name 속성만 있다면 name 속성으로도 폼 요소에 접근할 수 있다.

<form> 태그에도 name 속성이 있어야 하고, <form> 태그 안의 폼 요소에도 name 속성이 있어야 한다.

```
<form name="order">
  <fieldset>
    <legend>상품 정보</legend>
    <ul>
      <li>
        <label class="field" for="product">상품 : </label>
        <input type="text" class="input-box" id="product" name="product">
      </li>
      .....
    </ul>
  </fieldset>
  .....
```

상품 필드의 name값

document.order.product.value

form의 name값

상품 정보

상품 :

무선 마우스

갯수 :

1

요소

콘솔

Recorder

소스

네트워크

성능

메모리

애플리케이션

보안

Lighthouse

top

필터

> document.order.product.value

< '무선 마우스'

> |

## 폼 배열을 사용해 접근하기

폼 요소에 id나 class 속성뿐만 아니라 name 속성도 없다면 ??

DOM에서는 웹 문서 안에 있는 모든 요소를 배열 형태로 저장한다.

문서 안에 있는 <form> 태그는 모두 forms 라는 프로퍼티에 저장되어 있다.

```
document.forms
```

폼 안에 있는 요소들은 elements 속성에 역시 배열 형태로 저장된다.

```
document.forms[0].elements
```

document.forms

```
> document.forms
< ▼ HTMLCollection [form, order: form] ⓘ
  ▶ 0: form
  ▶ order: form
    length: 1
  ▶ [[Prototype]]: HTMLCollection
>
```

document.forms[0].elements

```
▶ 0: fieldset
▶ 1: input#product.input-box
▶ 2: input#prod-num.input-box
▶ 3: fieldset
▶ 4: input#order-name.input-box
▶ 5: input#order-tel.input-box
▶ 6: input#order-addr.input-box
▶ 7: button.order
▶ order-addr: input#order-addr.input-box
▶ order-name: input#order-name.input-box
▶ order-tel: input#order-tel.input-box
▶ prod-num: input#prod-num.input-box
▶ product: input#product.input-box
  length: 8
```

폼 배열을 써서 주문자 이름을 입력하는 필드에 접근하려면 (id, class, name 속성 모두 없다고 가정함)

```
<form name="order">
  <fieldset>
    <input type="text" class="input-box" id="product" >
    <input type="number" class="input-box" id="prod-num">
    </fieldset>
    <fieldset>
      <input type="text" class="input-box" id="order-name">
      <input type="text" class="input-box" id="order-tel">
      <input type="text" class="input-box" id="order-addr">
    </fieldset>
```

```
document.forms[0].elements[4]
```

```
> document.forms[0].elements[4]
< <input type="text" class="input-box" id="order-name"
    name="order-name">
>
```

# 선택 목록과 항목에 접근하기

19WgetForm.html을 웹 브라우저에서 열고 콘솔 창에서 연습합니다

```
<select name="major" id="major">
  <option>---- 학과 선택 ----</option>
  <option value="archi">건축공학과</option>
  <option value="mechanic">기계공학과</option>
  <option value="indust">산업공학과</option>
  <option value="elec">전기전자공학과</option>
  <option value="computer">컴퓨터공학과</option>
  <option value="chemical">화학공학과</option>
</select>
```

선택 목록에 접근하기

```
document.querySelector("#major")
```

선택 목록에 있는 항목에 접근하기

```
document.querySelector("#major").options
```



# 선택 목록과 항목에 접근하기

```
> document.querySelector("#major").options
< ▼ HTMLOptionsCollection(7) [option, option, option, opt
  ▶ 0: option
  ▶ 1: option
  ▶ 2: option
  ▶ 3: option
  ▶ 4: option
  ▶ 5: option
  ▶ 6: option
    length: 7
    selectedIndex: 0
  ▶ [[Prototype]]: HTMLOptionsCollection
>
```

selectedIndex :

선택 목록에서 선택한 항목의 인덱스 값

# 선택 목록과 항목에 접근하기

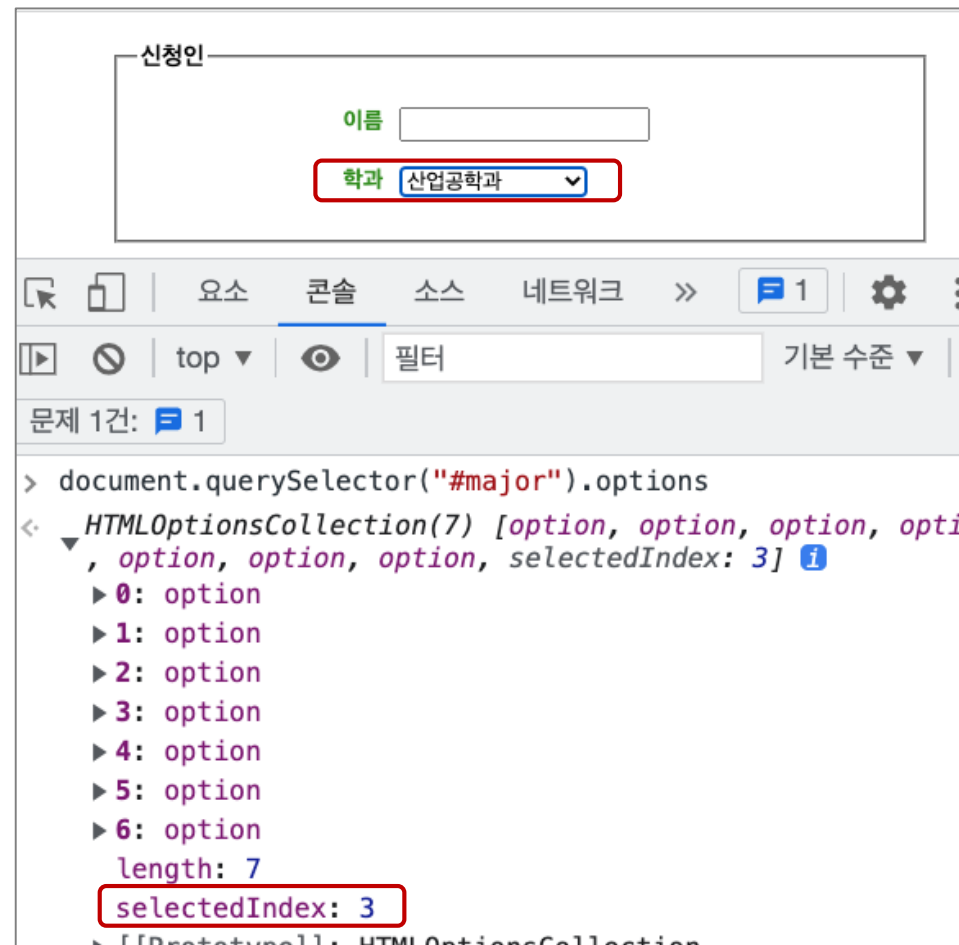
목록에서 '산업공학과'를 선택한 후  
콘솔 창에서 selectedIndex를 확인하면

```
document.querySelector("#major").options
```

선택된 항목의 인덱스 확인은 아래 두 가지 모두 가능

```
document.querySelector("#major").options.selectedIndex
```

```
document.querySelector("#major").selectedIndex
```



## (예) 선택한 항목 화면에 표시하기

19WgetForm.html, 19WjsWgetForm.js에서 연습

- 1) 선택 목록을 가져와 변수에 저장하고
- 2) 옵션의 selectedIndex를 확인한 후
- 3) 그 인덱스 값을 가진 옵션 항목의 내용을 가져온다

## (예) 선택한 항목 화면에 표시하기

19WjsWgetForm.js

```
const selectMenu = document.querySelector("#major");

function displaySelect() {
    let selectedText = selectMenu.options[selectMenu.selectedIndex].innerText;
    alert(`[${selectedText}]를 선택했습니다.`);
}

selectMenu.onchange = displaySelect;
```



화살표 함수로 바꾸면 ??

## (예) 선택한 항목 화면에 표시하기

19WjsWgetForm.js

```
const selectMenu = document.querySelector("#major");

selectMenu.onchange = () => {
  let selectedText = selectMenu.options[selectMenu.selectedIndex].innerText;
  alert(`[${selectedText}]를 선택했습니다.`);
}
```

# 라디오 버튼과 체크 박스에 접근하기

- 라디오 버튼이나 체크 박스는 `name` 을 사용해 버튼을 그룹으로 묶는다.  
(라디오 버튼이나 체크 박스는 하나의 그룹 안에서 항목을 선택하기 때문)
- 라디오 버튼과 체크박스는 `name` 값을 사용해 접근한다.
- 같은 `name`을 가진 항목이 많기 때문에 `RadioNodeList`라는 노드 리스트 형태로 저장됨.(배열과 비슷한 형태)
- 어떤 항목을 선택했는지 알려면 `checked` 속성이 있는지 체크  
(`checked` 속성은 HTML에서 라디오 버튼과 체크 박스에서 사용할 수 있는 속성)

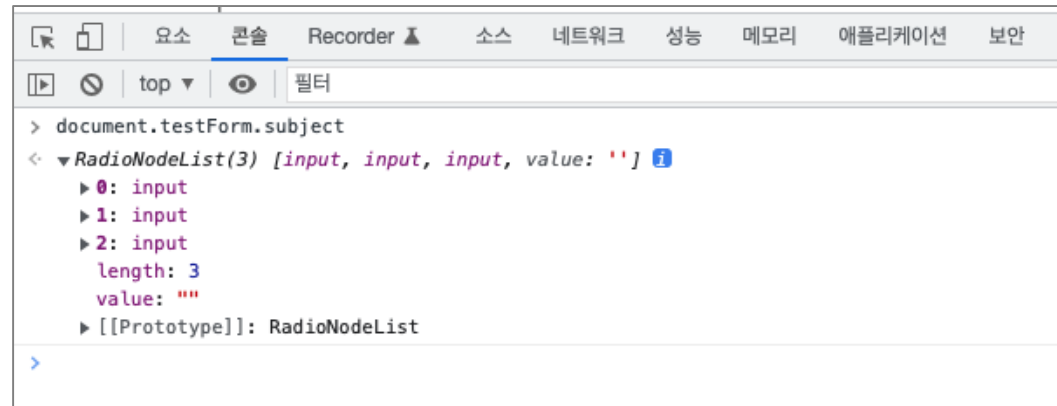
# 라디오 버튼에 접근하기

19WgetForm.html을 웹 브라우저에 열어놓고 콘솔 창에서 연습

```
<fieldset>
  <legend>신청 과목</legend>
  <p>이 달에 신청할 과목을 선택하세요.</p>
  <label><input type="radio" name="subject" value="speaking">회화</label>
  <label><input type="radio" name="subject" value="grammar">문법</label>
  <label><input type="radio" name="subject" value="writing">작문</label>
</fieldset>
```

`document.testForm.subject`   또는

`document.querySelector("form").subject`



# 체크 박스에 접근하기

19WgetForm.html을 웹 브라우저에 열어놓고 콘솔 창에서 연습

```
<fieldset>
<legend>메일링</legend>
<p>메일로 받고 싶은 뉴스 주제를 선택해 주세요</p>
<label><input type="checkbox" id="new" name="mailing" value="news">해외 단신</label>
<label><input type="checkbox" id="dialog" name="mailing" value="dialog">5분 회화</label>
<label><input type="checkbox" id="pops" name="mailing" value="pops">모닝팝스</label>
</fieldset>
```

document.testForm.mailing

또는

document.querySelector("form").mailing

```
> document.querySelector("form").mailing
< ▼ RadioNodeList(3) [input#new, input#dialog, input#pops, value: ''] ⓘ
  ▶ 0: input#new
  ▶ 1: input#dialog
  ▶ 2: input#pops
    length: 3
    value: ""
  ▶ [[Prototype]]: RadioNodeList
```



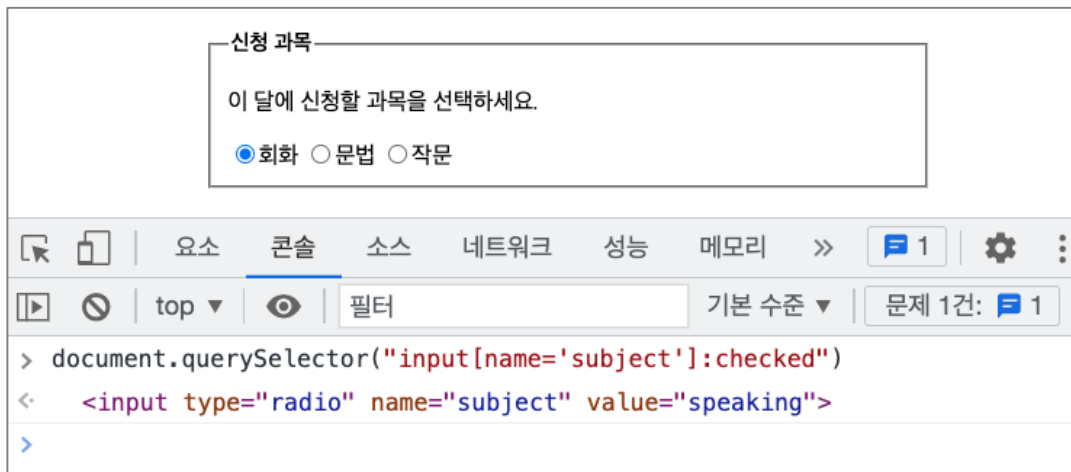
# checked 속성 사용하기

- 라디오 버튼이나 체크 박스 항목 중에서 선택하면 해당 항목에 checked 속성이 추가됨.
- 즉, checked = true 상태가 됨

19\getForm.html 문서를 열고 '회화'를 선택한 상태에서

콘솔 창에 name="subject"인 요소 중에서 선택된 것을 찾는 소스를 입력해 보자

```
document.querySelector("input[name='subject']:checked")
```



19\getForm.html 문서를 열고 '메일링' 체크 박스에서 2가지 박스를 선택하고  
콘솔 창에서 name="mailing"인 요소 중에서 선택된 것을 찾는 소스를 입력해 보자.

주의: 체크박스는 2개 이상 선택 가능하므로 `querySelectorAll()` 사용해야 함!

```
document.querySelectorAll("input[name='mailing']:checked")
```

