# Form

## #01. <form>요소의 객체 획득하기

1) id속성값으로 접근하는 경우

```
// form의 id속성값이 `form_id`인 경우
const myform = document.getElementById("form_id");
```

2) name속성값으로 접근하는 경우

```
// form의 name속성값이 `form_name`인 경우
const myform = document.form_name;
```

## #02. 획득한 <form> 안의 <input>요소 접근

1) id속성값으로 접근하는 경우

```
const hello = document.getElementById("input_id");
```

2) name속성값으로 접근하는 경우

```
const myform = document.form_name;
const hello = myform.input_name;
```

name속성값을 사용하는 경우 아래와 같이 축약도 가능함.

```
const hello = document.form_name.input_name;
```

## #03. 획득한 <input>요소의 입력값 처리

input요소의 value 속성을 사용하여 입력값을 조회하거나 설정할 수 있다.

```
const myform = document.form_name;
const hello = myform.input_name;
hello.value = "hello world";        // 값의 설정
const msg = hello.value;            // 값의 조회
```

name속성값을 사용하는 경우 아래와 같이 축약도 가능함.

```
document.form_name.input_name.value = "hello world";
const msg = document.form_name.input_name.value;
```

획득한 <input>요소에 대한 값의 입력여부 조회

```
if (!document.form_name.input_name.value) {
    // 값이 입력되지 않은 경우의 처리
    alert("내용을 입력하세요.");
    document.form_name.input_name.focus();
    return;
}
```

# #04. 선택 가능한 요소의 제어

## 1) 드롭다운의 제어

- <select>태그로 표현되는 드롭다운 요소는 그 자체가 배열이며 <option>태그들이 배열의 요소가 된다.
- 드롭다운 객체에 대한 selectedIndex는 선택된 요소의 배열 인덱스를 의미

```
const dropdown = document.form_name.select_name;

// 특정 위치 강제 선택
dropdown.selectedIndex = 2;

// 현재 선택된 요소의 위치
const choice_index = dropdown.selectedIndex;

// 현재 선택된 요소의 value값
const choice_value = dropdown[choice_index].value;
```

주로 첫 번째 <option>요소는 선택을 요구하기 위한 안내문이 표시되는 경우가 많기 때문에 selectedIndex값이 0인 경우는 선택한 것으로 간주하지 않는다.

```
if (dropdown.selectedIndex == 0) {
    ... 선택되지 않은 경우의 처리 ...
}
```

## 2) checkbox, radio의 제어

name속성이 동일한 요소들끼리 하나의 배열로 그룹화 됨.

```
const foo = document.form1.myradio[0].value;
```

checked 프로퍼티를 사용하여 선택 여부를 조회하거나 선택 상태를 강제 설정 할 수 있음.

```
// 선택 여부 검사
const foo = document.form1.myradio[0].chekcked;
if (foo) {
    ... 선택된 경우의 처리 ...
}

// 강제 선택
document.form1.myradio[0].checked = true;
```

배열이라는 특성 때문에 반복문을 통한 제어가 가능.

```
const checkbox = document.form_name.checkbox_name;
const count = 0;    // 선택된 항목의 수

for (const i=0; i<checkbox.length; i++) {
    if (checkbox[i].checked) {
        count++;
    }
}
```

## #05. <form>의 reset, submit 처리

### 1) reset

```
document.form_name.reset();
```

### 2) submit

- <form>요소의 action속성에 지정된 페이지로 사용자의 입력내용을 전송하는 기능.
- <input type="submit"/>요소를 클릭하거나 아래의 자바스크립트 구문의 호출을 통해서 구현된다.

```
document.form_name.submit();
```

### 3) submit 이벤트

<form>요소에 onsubmit 이벤트 적용후 데이터 전송을 방지하기 위한 e.preventDefault() 처리가 필요하다.

입력값을 검사한 후 코드 하단부에서 강제로 submit() 처리한다.

```
document.querySelector("폼요소에 대한 selector").addEventListener("submit", e
=> {
    e.preventDefault();
    ...
    e.currentTarget.submit();
});
```

> 이벤트 리스너에 전달된 콜백함수 안에 에러가 있을 경우 그 이후 부분은 실행되지 않기 때문에 `submit`이 동작하게 되어 페이지가 새로고침되는 현상이 발생한다. 이 경우 한 블록씩 코드를 지워가면서 에러가 발생하는 위치를 찾아야 한다. 이를 방지하기 위해 `e.preventDefault()`를 적용해야 한다.

## #06. 자주 사용되는 정규표현식

숫자 모양에 대한 형식 검사

```
/^[0-9]*$/
```

영문으로만 구성되었는지에 대한 형식 검사

```
/^[a-zA-Z]*$/
```

한글로만 구성되었는지에 대한 형식 검사

```
/^[ㄱ-ㅎ가-힣]*$/
```

영문과 숫자로만 구성되었는지에 대한 형식 검사

```
/^[a-zA-Z0-9]*$/
```

한글과 숫자로만 구성되었는지에 대한 형식 검사

```
/^[ㄱ-ㅎ가-힣0-9]*$/
```

이메일 형식인지에 대한 검사. 아이디@도메인의 형식을 충족해야 한다.

```
/^([\w-]+(?:\.[\w-]+)*)@((?:[\w-]+\.)*\w[\w-]{0,66})\.([a-z]{2,6}(?:\.[a-
z]{2})?)$/i
```

-없이 핸드폰번호인지에 대한 형식검사.

```
/^01(?:0|1|[6-9])(?:\d{3}|\d{4})\d{4}$/
```

-없이 전화번호인지에 대한 형식검사. 각 부분에 대한 자리수도 충족시켜야 한다.

```
/^\d{2,3}\d{3,4}\d{4}$/
```

-없이 주민번호에 대한 글자수 및 뒷자리 첫글자가 1~4의 범위에 있는지에 대한 검사

```
/^\d{6}[1-4]\d{6}/
```