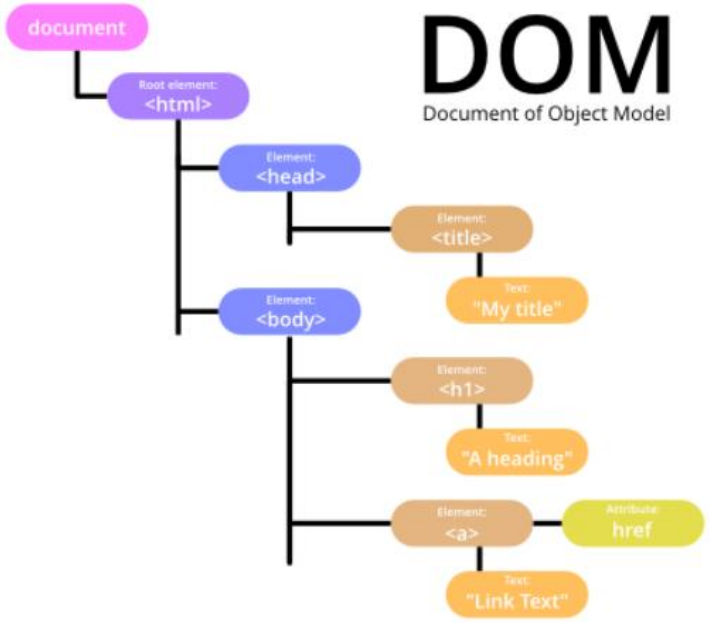
자바스크립트

문서-객체 모델

목차

문서 객체

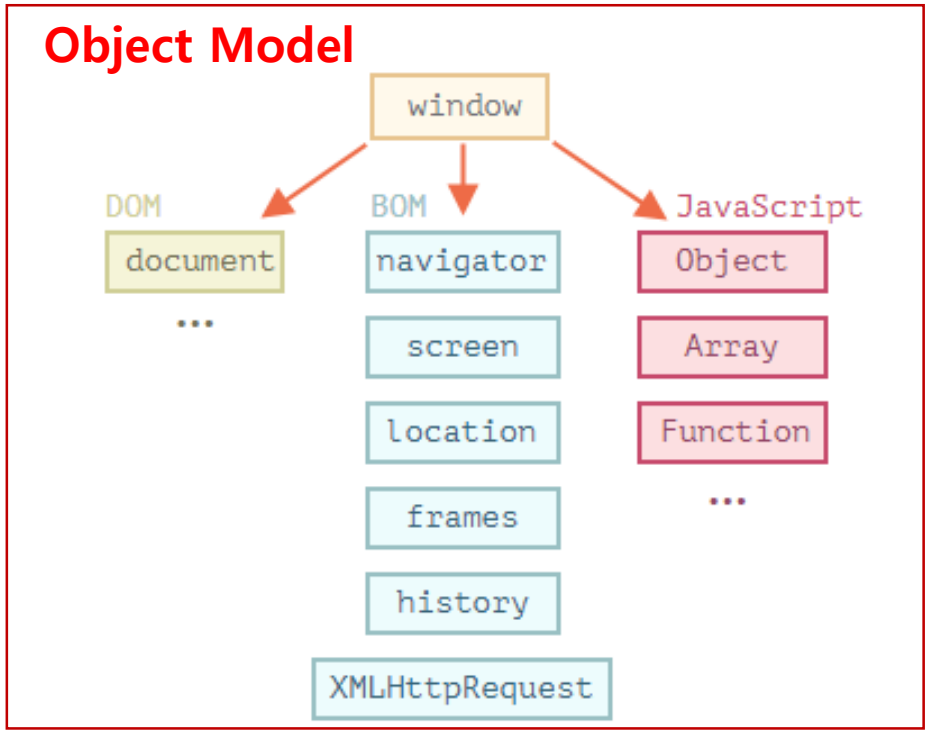
- 문서 객체
  - 웹 브라우저는 HTML 페이지에 있는 Element들은 트리 형식으로 메모리에 로딩한다.
  - 이때 각 노드를 “document object”라 한다.
  - 웹 브라우저는 이러한 문서객체를 접근 할 수 있도록 API(데이터+기능)를 제공하는데 이를 “문서 객체 모델”(DOM)이라고 한다.
  - JS는 DOM을 호출/제어 하기 위한 가장 보편적 언어이다.



문서 객체

- 문서객체, 브라우저 객체, 오브젝트모델
  - DOM : HTML 문서 객체의 집합
  - BOM : 브라우저 자체를 제어하는 객체 집합
  - JS : 자바스크립트를 위한 객체 집합

  - Object Mode : DOM+DOM+JS



문서 객체

- DOM을 제어하기 위한 일반적인 방법
  1> 제어 대상(객체)을 찾는다.
  2> 명령을 내린다.
  3> 브라우저가 처리한다.

ex 1> 이미지 객체를 찾는다.
2> 이미지의 사이즈를 변경한다.
3> 브라우저가 처리한다.

문서객체

- Object Model의 구성
  - console에서 확인하면 window 부터 하위로 구성됨을 알 수 있다.

```
> window
◂ ▶Window {window: Window, self: Window, document: document, name: '', location: Location, …}
> window.document
◂ ▶#document
> window.browser
◂ undefined
> window.location
◂ ▶Location {ancestorOrigins: DOMStringList, href: 'about:blank', origin: 'null', protocol: 'about:', host: '', …}
> window.Object
◂ ƒ Object() { [native code] }
> window.Array
◂ ƒ Array() { [native code] }
```

  - BOM을 제어하기
    window.location.href='http://www.naver.com'

기본적인 문서 객체 제어

- DOMContentLoaded 이벤트
  - 초기 HTML 문서를 완전히 불러오고 분석했을 때 발생하는 이벤트
  - JS 엔진은 <script>를 순차적으로 (위에서 아래로) 수행
  - 즉 JS엔진은 html문서가 다 로딩되기 전에도 수행 될 수 있음.
  - 객체 제어를 위해서는 문서가 다 로딩되고 나서 제어를 시작해야함 (DOM구조가 다 만들어지고 나서 제어를 수행)
  - DOMContentLoaded는 문서가 완전하게 로딩되고 난 후 이벤트 발생

기본적인 문서 객체 제어

- DOMContentLoaded 이벤트의 필요성

ERROR 발생, body를 읽기 전이기 때문에 에러

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        document.body.innerHTML += '<h1>Hello world</h1>';
    </script>
</head>
<body>

</body>
</html>
```

정상 작동, body를 읽었기때문에 정상동작

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <script>
        document.body.innerHTML += '<h1>Hello world</h1>';
    </script>
</body>
</html>
```

기본적인 문서 객체 제어

- DOMContentLoaded 이벤트 처리.

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        const callbackDOMLoaded = function()
        {
            document.body.innerHTML += "<h1>Hello World</h1>"

        }

        window.addEventListener('DOMContentLoaded', callbackDOMLoaded)

    </script>
</head>
<body>
</body>
</html>
```

DOMContentLoaded를 사용하거나 body의 마지막에 둔다.

문서 객체 선택

- 문서 객체의 선택방법
  - 상위 Document Object의 선택

  document.head
  document.body
  document.title

  - 특정 Document Object의 선택

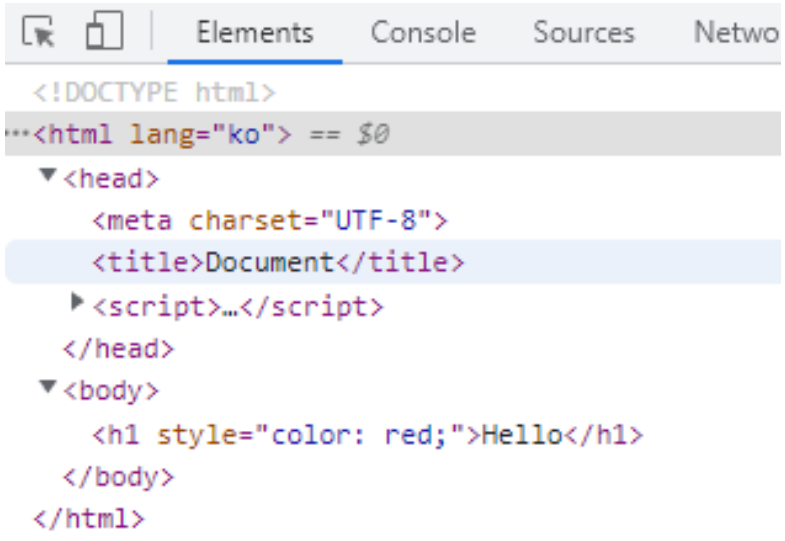
  document.querySelector(selector); // 단일 선택
  document.querySelectorAll(selector);// 복수 선택
  **** 여기서 selector는 CSS selector와 같다.**

문서 객체 선택

- querySelector()

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        const callbackDOMLoaded = function()
        {
            const header = document.querySelector('h1');
            header.style.color = 'red';
        }

        window.addEventListener('DOMContentLoaded', callbackDOMLoaded)
    </script>
</head>
<body>
    <h1>Hello</h1>
</body>
</html>
```



문서 객체 선택

- querySelectAll()

```
<script>
    const callbackDOMLoaded = function()
    {
        const header = document.querySelectorAll('h1');
        header.forEach((element)=>{
            element.textContent = 'HEAD';
            element.style.color = 'white';
            element.style.backgroundColor = 'red';
            element.style.padding = '2px';

        })
    }

    window.addEventListener('DOMContentLoaded', callbackDOMLoaded)
</script>
```

```
<body>
    <h1></h1>
    <h1></h1>
    <h1></h1>
</body>
```

결과

HEAD

HEAD

HEAD

연습문제

- body 내용이 아래와 같을때 첫번째 div에 <h1>Hello</h1>, 두번째 div에 <h1>World</h1> 나오게 코딩하시오

```
<div></div>
<div></div>
```

- body 내용이 아래와 같을때 cls_imgcat에만 200*200이미지를 출력하시오

이미지 소스 : http://www.placekitten.com/200/200

```
<img class="cls_imgcat">
<img class="cls_imgcat">
<img class="cls_imgcat">
<img>
<img>
```

문서 객체 속성

- 속성의 조작
  - 객체.setAttribute() :속성을 설정
  - 객체. getAttribute() : 속성을 구함.

```
<body>
    <h1 style="color: red">Hello World</h1>

    <script>
        const head1 = document.querySelector('h1');
        let style = head1.getAttribute('style');
        console.log(style);

    </script>
</body>
```

문서 객체 속성

- 글자 조작
  - 객체.textContent : 입력된 문자열을 해석 없이 그대로 입력
  - 객체.innerHTML : 입력된 문자열을 해석 하여 HTML형식으로 입력

  - Content를 수정하거나 삽입할경우는 textContent, html코드를 입력할 때는 innerHTML을 사용.

```
<body>
    <div id="d1"></div>
    <div id="d2"></div>

    <script>
        const div1 = document.querySelector('#d1');
        const div2 = document.querySelector('#d2');

        div1.innerHTML = '<h1>Hello</h1>';
        div2.textContent = '<h1>Hello</h1>';
    </script>
```

문서 객체 속성

- 스타일의 조작
  - CSS에서 스타일 사용시 '–'를 사용 할 수 있으나 자바스크립트에서는 사용할 수 없음.
  - 자바스크립트에서는 카멜스타일로 스타일을 처리

CSS	JS style
background-color	backgroundColor
text-align	textAlign

  - h1.style['background-color'] 형태로 계산된 문자열을 사용할 수 있다.

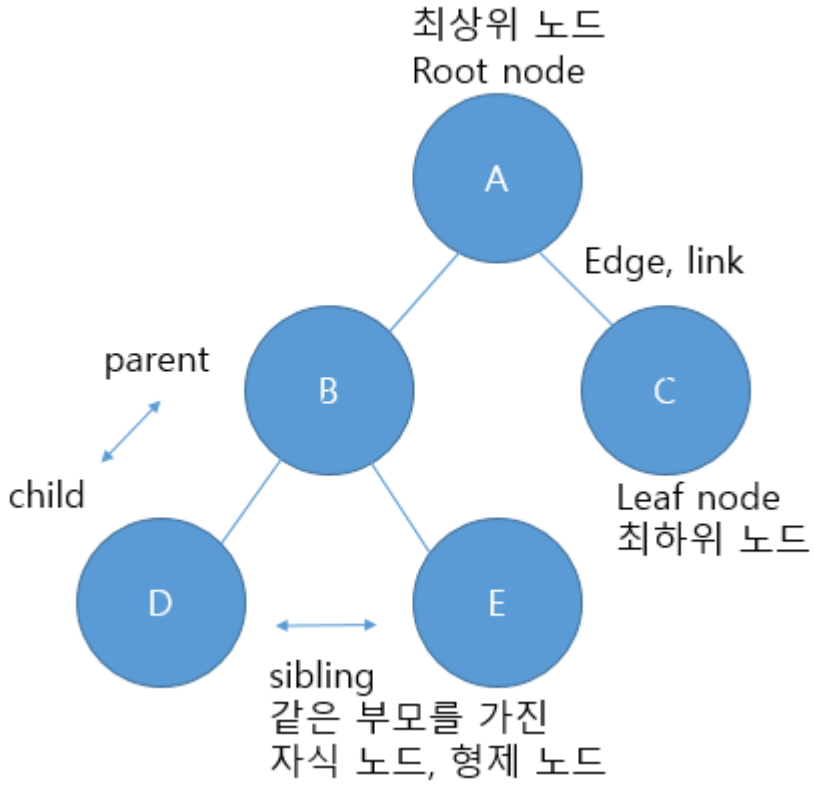
문서 객체 속성

- 문서 객체의 생성 및 추가
  - 문서객체의 생성 : document.createElement()
  - 문서객체의 추가 : 객체.appendChild()

```
<body>
    <div id="d1"></div>

    <script>
        const h1tag = document.createElement('h1');
        const d1tag = document.querySelector('#d1');

        h1tag.textContent = 'Hello';
        d1tag.appendChild(h1tag);
    </script>
</body>
```



문서 객체 속성

- 문서 객체의 제거
  - 문서객체의 제거 : 부모객체.removeChild(자식객체)

```
<body>
    <div id="d1"></div>

    <script>
        const h1tag = document.createElement('h1');
        const d1tag = document.querySelector('#d1');

        h1tag.textContent = 'Hello';
        d1tag.appendChild(h1tag);

        //3초 후에 제거한다.
        setTimeout(()=>{ d1tag.removeChild(h1tag); }, 3000);
    </script>
</body>
```

이벤트

- 이벤트의 개념
  - Event : 어떤 조건이 만족 될 때 발생되는 신호
  - message : 이벤트의 값.
  - Event Handler : 이벤트 발생시 호출되는 함수 event listene라고도 함
  - 자바스크립트의 이벤트 처리시스템은 Callback 함수를 통해 구현된다.
  - 이는 윈도우시스템의 이벤트 처리 시스템과 거의 유사하다.

이벤트

- 이벤트 처리 방식
  - 표준방식 : addEventListener()
  - 고전방식 : 태그를 이용

  - 고전방식은 ES5에서 사용했던 방식을 말하며 현재는 표준방식(표준 이벤트 처리 모델) 사용.

이벤트

- 이벤트 처리방식 - 표준방식
  - 이벤트 연결 : addEventListener() 및 콜백함수로 처리한다.

```
<body>
    <h1>Click Count 0</h1>

    <script>
        let count = 0;
        const h1tag = document.querySelector('h1');

        h1tag.addEventListener('click', (event)=>{
            count++;
            h1tag.textContent = 'Click Count ' + count;
        })

    </script>
</body>
```

이벤트

- 이벤트 처리방식 - 표준방식
  - 이벤트 제거 : 이벤트 핸들러를 임시객체로 사용불가

```
<body>
    <h1>Click Count 0</h1>

    <script>
        let count = 0;
        const h1tag = document.querySelector('h1');

        const OnClick = function(event){
            count++;
            h1tag.textContent = 'Click Count ' + count;

            if (count === 10)
                h1tag.removeEventListener('click', OnClick);
        }

        h1tag.addEventListener('click', OnClick);
    </script>
</body>
```

이벤트

- 이벤트 처리방식 - 고전방식

```
<body>
    <h1 onClick="myFunction()">Test</h1>

    <script>
        const h1 = document.querySelector('h1');

        function myFunction()
        {
            alert("Click");


        }
    </script>
</body>
```

이벤트

- 키보드 이벤트
  - 키보드 이벤트의 종류
    - keydown : 키가 눌렸을 때
    - keypress : 키가 입력되었을 때 (Shift, Ctrl, Fn키등 특수키 제외)
    - keyup : 키가 떨어질 때
  - 키보드 이벤트의 처리
    - event객체에서 정보를 얻음.
    - 일반적으로 Keyup에서 처리
    - keydown으로 처리시 한국어나 중국어의 경우 처리를 못하는 경우가 있음. (글자를 조합하기 때문에 이벤트를 처리 안할 수 있음)

이벤트

- 키보드 이벤트

```
<body>
    <h1></h1>
    <textarea></textarea>

    <script>
        const tarea = document.querySelector('textarea');
        const h1 = document.querySelector('h1');

        tarea.addEventListener('keydown', (event)=>{
            const len = tarea.value.length;
            h1.textContent = `글자수 : ${len}`;
        })
    </script>
</body>
```

이벤트

- 키코드
  - code : 입력한 키
  - keyCode : 키의 실제값
  - altKey, ctrlKey, shiftKey : 알트, 컨트롤, 쉬프트

```
<body>
    <h1></h1>

    <script>
        const h1 = document.querySelector('h1');
        const printKey = function(event)
        {
            let output = '';
            output += `code = ${event.code} <br>`;
            output += `keyCode = ${event.keyCode} <br>`;
            output += `alt = ${event.altKey} <br>`;
            output += `Ctrl = ${event.ctrlKey} <br>`;
            output += `Shift = ${event.shiftKey} <br>`;

            h1.innerHTML = output;

        }
        document.addEventListener('keydown', printKey);
        document.addEventListener('keyup', printKey);
    </script>
</body>
```

이벤트

- 키보드 이벤트 예제

```
<body>
    <h1>★</h1>
    <script>
        const [left, up, right, down] = [37, 38, 39 ,40];

        const star = document.querySelector('h1');
        star.style.position = 'absolute';

        let x = 0;
        let y = 0;
        let offset = 20;
        const print = function(_x, _y)
        {
            star.style.left = `${_x * offset}px`;
            star.style.top  = `${_y * offset}px`;

        }

        print(x, y);

        const moveStar = function(event)
        {
            switch (event.keyCode)
            {
                case left:  x-= 1; break;
                case up:    y-= 1; break;
                case right: x+= 1; break;
                case down:  y+= 1; break;
                default: return; break;
            }

            print(x, y);

        }

        document.addEventListener('keydown', moveStar);

    </script>
</body>
```

이벤트

- 이벤트 수신 객체
  - 이벤트 핸들러에서 이벤트를 받은 객체가 누구인지 알아내는 기법
    - object
    - this
    - event.currentTarget

```
<body>
    <h1></h1>
    <textarea></textarea>
    <script>
        const tarea = document.querySelector('textarea');
        const h1 = document.querySelector('h1');

        tarea.addEventListener('keyup', (event)=> {
            const len = tarea.value.length;
            h1.textContent = `글자수 : ${len}`;
        });
    </script>
</body>
```

- object : tarea 로 직접 접근한다.

이벤트

- 이벤트 수신 객체

```
<body>
    <h1></h1>
    <textarea></textarea>
    <script>
        const tarea = document.querySelector('textarea');
        const h1 = document.querySelector('h1');

        tarea.addEventListener('keyup', function(event){
            const len = this.value.length;
            h1.textContent = `글자수 : ${len}`;
        });
    </script>
</body>
```

- this : this는 tarea의 레퍼런스 값.
- 만약 화살표 함수를 사용하면 this를 사용할 수 없다.

이벤트

- 이벤트 수신 객체

```
<body>
    <h1></h1>
    <textarea></textarea>
    <script>
        const tarea = document.querySelector('textarea');
        const h1 = document.querySelector('h1');

        tarea.addEventListener('keyup', (event)=>{
            const len = event.currentTarget.value.length;
            h1.textContent = `글자수 : ${len}`;
        });
    </script>
</body>
```

- event.currentTarget : tarea의 레퍼런스 값.
- 화살표 함수, 일반함수 모두 사용 가능

form 처리

- form의 값 얻어오기

```
<script>
document.addEventListener('DOMContentLoaded', ()=>{

    const input = document.querySelector('input');
    const button = document.querySelector('button');
    const h3 = document.querySelector('h3');

    const onButtonClick = function(event)
    {
        const radius = Number(input.value);

        if (isNaN(radius))
        {
            h3.textContent = '숫자를 입력하세요.'
            return;
        }

        h3.textContent = `원의 넓이 = ${(radius * radius * 3.14).toFixed(2)}`

    }

    button.addEventListener('click', onButtonClick);

})
</script>
```

```
<body>
    <h1>원의 넓이</h1>
    <label>반지름 : <input type="text"></label>
    <button>계산</button>
    <h3></h3>

</body>
```

- input.value를 통해 form의 값을 얻어온다.

form 처리

- select 처리
  - options객체의 정보를 이용하여 현재 선택한 option, index를 구한다.

```
<script>
document.addEventListener('DOMContentLoaded', ()=>{
    const select = document.querySelector('select');
    const p1 = document.querySelector('p');

    select.addEventListener('change', (event)=>{
        const options = event.currentTarget.options;
        const index = event.currentTarget.options.selectedIndex;

        p1.textContent = `선택 : ${options[index].textContent}`;

    })

})
</script>
```

```
<body>
    <select>
        <option>순대국밥</option>
        <option>뼈해장국</option>
        <option>돈까스</option>
        <option>다이어트</option>
    </select>
    <p>선택 : 순대국밥</p>
</body>
```

form 처리

- multi-select 처리

```
<script>
document.addEventListener('DOMContentLoaded', ()=>{
    const select = document.querySelector('select');
    const p1 = document.querySelector('p');

    select.addEventListener('change', (event)=>{
        const options = event.currentTarget.options;
        const list=[];

        for (const option of options)
        {
            if (option.selected)
                list.push(option.textContent);

        }

        p1.textContent = `선택 : ${list.join(', ')}`;

    })

})
</script>
```

```
<body>
    <select multiple>
        <option>순대국밥</option>
        <option>뼈해장국</option>
        <option>돈까스</option>
        <option>다이어트</option>
    </select>
    <p></p>
</body>
```

- options 속성에는 forEach() 와 같은 callback기능이 없음
- 따라서 loop문으로 처리

form 처리

- 체크박스

```
<script>
document.addEventListener('DOMContentLoaded', ()=>{
    const checkbox = document.querySelector('input');
    const p1 = document.querySelector('p');
    let msg = '';

    checkbox.addEventListener('change', (event)=>{
        if (event.currentTarget.checked)
            msg = 'Checked';
        else
            msg = 'Uncheked';

        p1.textContent = msg;
    })

})
</script>
```

```
<body>
    <label>Checkbox<input type="checkbox"></label>
    <p>Uncheked</p>

</body>
```

form 처리

- 라디오버튼 처리

```
<script>
document.addEventListener('DOMContentLoaded', ()=>{
    const radios = document.querySelectorAll('input[name=gender]');
    const p1 = document.querySelector('p');

    radios.forEach((radio)=>{
        radio.addEventListener('change', (event)=>{
            if (event.currentTarget.checked)
            {
                p1.textContent = `${event.currentTarget.value}`;

            }
        });
    });

});
</script>
```

```
<body>
    <label>선택안함<input type="radio" name="gender" value="nogender"></label>
    <label>남성<input type="radio" name="gender" value="male"></label>
    <label>여성<input type="radio" name="gender" value="female"></label>
    <p></p>

</body>
```

연습문제

- 이메일을 입력받아 id와 도메인을 추출하는 코드를 작성하시오
  단 이메일은 prompt()로 입력받음.

- 인치를 센치로 바꾸는 코드를 작성하시오

```
<body>
    <input type="text">inch<br>
    <button>계산</button>
    <p>결과 : </p>
</body>
```

연습문제

- 이메일 형식인지 검사하는 코드를 작성하시오. 단 검사는 키를 누를때마다(keyup) 검사해야 함
  - 이메일형식
    - @가 있어야 함.
    - @ 뒤에 . 이 있어야 함

```
<body>
    <input type="text">
    <p>이메일 형식 : NO </p>
</body>
```

- 체크박스를 클릭하면 “x초”라고 초단위로 시간이 흐르고 다시 클릭하여 체크를 풀면 중지하는 코드를 작성하시오