{제어를 통해 웹페이지를 풍부하게 해보자!}

-배열, 함수, 객체

배열(array)

<같은 타입의 여러 변수를 하나의 묶음으로 다루는 것>

Javascript 배열

Java 배열

- 동일한 데이터 자료형만 저장 가능

- 배열의 크기가 고정적

- 정해진 배열의 크기만큼만 데이터를 추가 할 수 있음.

VS

Javascript 배열

- **다양한 데이터 자료형** 저장 가능

- 배열의 크기가 **가변적**

- 정해진 크기를 넘어서 데이터를 추가하면 **자동으로 저장공간 할당**

Javascript 선언&생성

- 배열 선언방식

```
let nameList = [];
let nameList = new Array();
```

- 배열 생성방식

```
let nameList = [“David”,”Will”,”Jhon”,”Sam”];
let nameList = new Array(“David”,”Will”,”Jhon”,”Sam”);
```

Javascript 배열접근

- 반드시 배열 생성 후 접근

```
let nameList;
nameList[2] = 3;
```

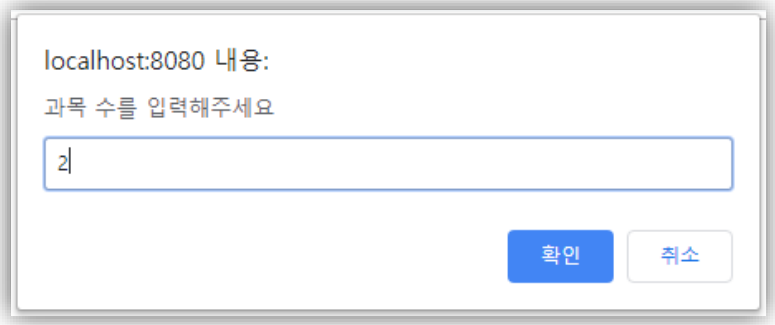
```
Uncaught TypeError: Cannot set property '0' of undefined
```

-> nameList가 정의되어 있지 않으므로 오류발생

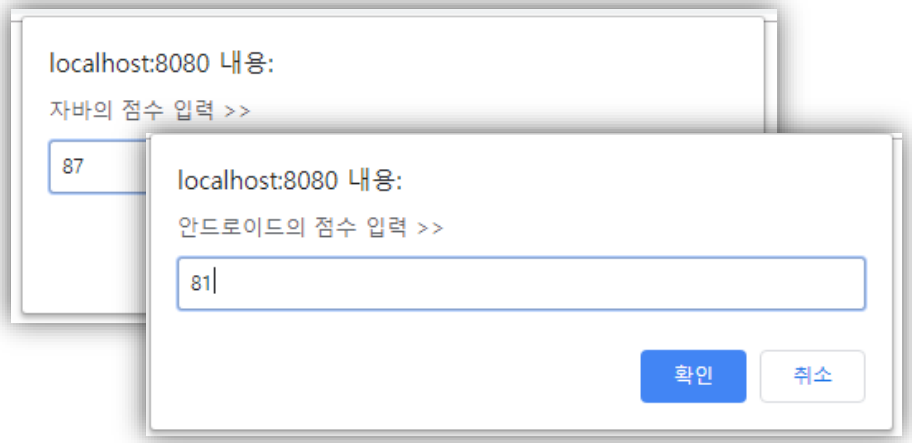
- 배열 변수명과 [] 사이에 원소의 인덱스를 적어 접근

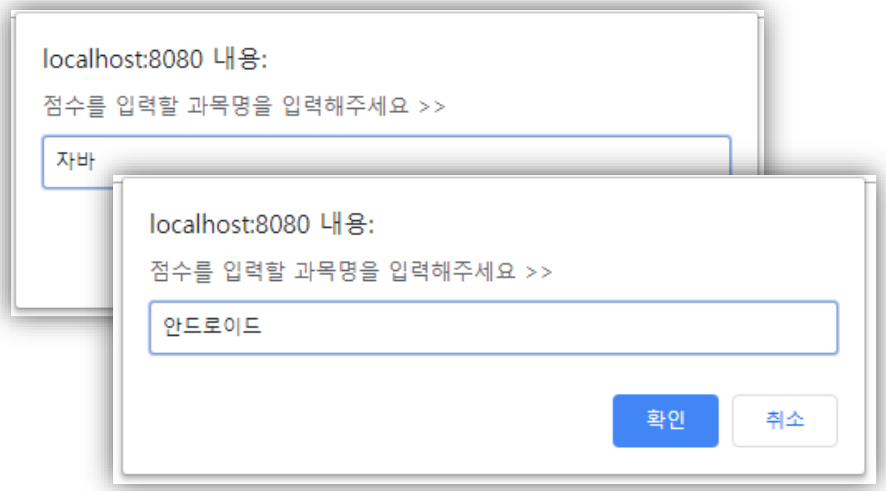
- 배열의 인덱스는 **0부터 시작**

Javascript 배열 실습



실행화면

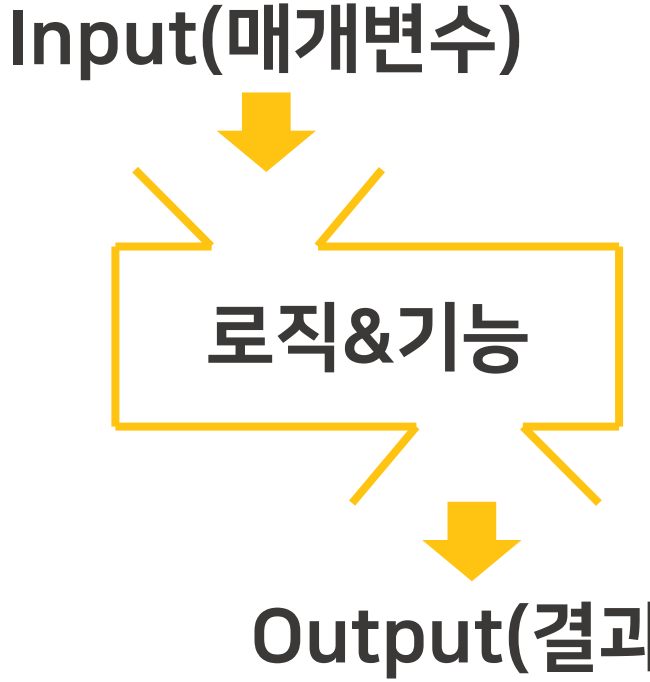




자바	안드로이드	총합	평균
87	81	168	84

결과화면

특정 기능을 수행하는 소스 코드를 하나로 묶어
필요할 때마다 호출하여 사용하기 위한 구조



"안녕하세요?"

alert()

Javascript 함수 사용 목적

```
<script type="text/javascipt">
        function 함수명 (){
            로직 & 기능구현을 위한 코드작성
        }
        함수명(); //함수호출
</script>
```

<Function 구조>

- 어떠한 실행코드를 묶어서 실행하기 위함
- 중복되는 코드 최소화
- 실행코드 블록화 => 코드조각화

호이스팅(Hoisting)

<함수 안에 있는 선언들을 모두 끌어올려
해당 함수 유효 범위의 최상단에 선언하는 것>

함수선언문(Function Declarations)

```
function 함수명 (매개변수1, 매개변수2,····){
        로직 & 기능구현을 위한 코드작성
}
함수명(입력값1, 입력값2, ····);
```

함수표현식(Function Expressions)

```
var fun1 = function (매개변수1, 매개변수2,····){
        로직 & 기능구현을 위한 코드작성
}
함수명(입력값1, 입력값2, ····);
```

새롭게 등장한 함수선언 방법

Arrow Function(ES6)

- function키워드 대신 화살표(=>)를 사용함
- this키워드 사용시 상위 scope의 this(Lexical this)를 가리킴

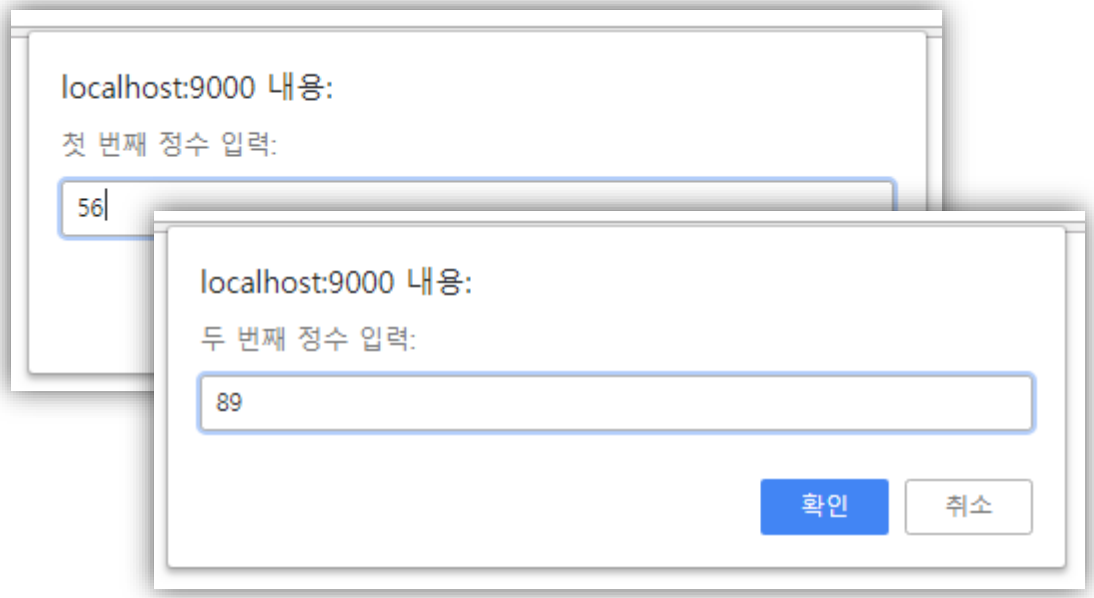
```
const 함수명 = (매개변수) => { 로직 }
```

Arrow function(ES6)

```
//es5
var add = function(a,b){
    return a+b;
}
//es6 - arrow function
const add = (a,b) => {
    return a + b;
}

//매개변수가 1개라면 소괄호도 생략할수있다.
const square = x => { return x*x };

//한줄로 작성 가능한 경우 중괄호와 return 키워드도 생략 가능하다.
const square = x => x*x;
const add = (a,b) => a+b;

//매개 변수가 없는 경우
() => { return { value : 1 }; }

//객체를 함수의 몸체와 구분해주기 위해 소괄호를 사용한다
() => ( { value : 1 } );
```

Javascript 함수 실습

두 정수를 입력받아 합을 구하는 addNumber() 함수를 작성하시오.



Javascript 함수 특징

1. 매개변수와 입력값의 데이터 타입이 동일한지 검사하지 않는다.

   -> why? 데이터타입을 지정하지 않는다.

```
<script>
        var num1 = 1;
        var num2="2";

        addNum(num1,num2);

        function addNum (n1, n2){
                alert(n1+n2);
        }
</script>
```

Javascript 함수 특징

2. 매개변수와 입력값의 개수가 같은지 확인하지 않는다.

-> why? 내부적으로 arguments 객체가 호출되어 인자들을 배열 형태로 저장한다.

```
<script>
        var num1 = 1;
        var num2 = 2;

        function addNum (n1, n2){ alert(n1+n2); }

        addNum(1,2);      -->    1 2
        addNum(1,2,3);    -->    1 2
</script>
```

Javascript 함수 특징

3. 입력값의 개수가 매개 변수의 개수보다 적다면 매개 변수의 값은 undefined로 설정된다.

```
<script>
        var num1 = 1;
        var num2 = 2;

        function addNum (n1, n2){ alert(n1+n2); }

        addNum();      -->      undefined undefined
        addNum(1);     -->      1 undefined
</script>
```

Javascript 함수 return

- 함수가 결과값을 반환하는 경우에는 함수 내에 반드시 return문을 사용
- 배열이나 객체를 포함한 모든 타입의 값을 반환 가능

```
<script>
        function 함수명 (){
                로직 & 기능구현을 위한 코드작성;
                return 반환값;
        }
</script>
```

객체(Object)

객체 지향 프로그래밍에서 **데이터(속성)**과
데이터에 관련되는 **동작(절차,방법,기능)**을 모두 포함한 개념

객체(Object) 기본 구조

```
<script>
    let 객체명 = {
        속성명1 : 값1,
        속성명2 : 값2,
        속성명3 : 함수(){
                    //기능구현
        }
                ...
    };
</script>
```

- 객체는 property(속성)과 method(기능)으로 구성
- 객체 내에는 기본데이터타입, Array, Object 등 데이터를 담을 수 있음
- 객체 내 데이터를 접근하는 방법은 마침표(.) 이용

객체(Object) 생성 방법

```
<script>
      //객체 생성
      let 객체명 = {};

      //속성추가
      객체명.속성1 = 값1;
      객체명.속성2 = 값2;
      객체명.속성3 = 함수(){
                  //기능구현
      }
</script>
```

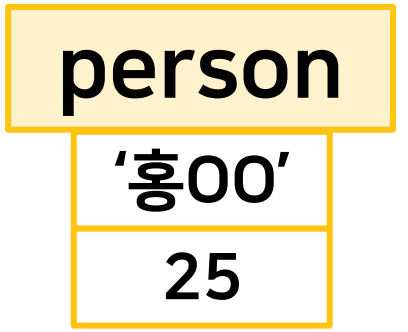
```
<script>
      let 객체명 = {
            속성명1 : 값1,
            속성명2 : 값2,
            속성명3 : 함수(){
                        //기능구현
            }
                  ...
      };
</script>
```

ex06obj_js.html

객체(Object) 예시1)

– 이름이 '홍OO'이고 나이는 25살인 person객체를 생성하시오.

```
<script>
    //object 객체 생성
    let person = {};

    //속성(property) 추가
    //객체명.프로퍼티
    person.name = '홍OO';
    person.age = 25;
</script>
```



<Javascript엔진 내 메모리 할당>

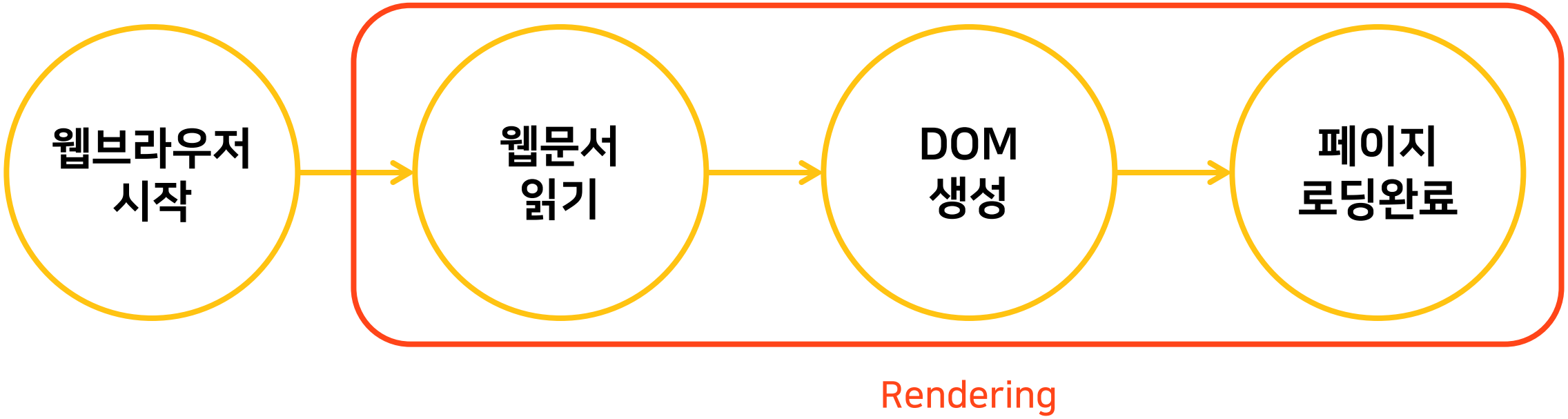
DOM

(Document Object Model)

HTML문서의 요소를 효과적으로 다루기 위하여
모든 요소를 분리하고 상하관계를 설정한 후 배치한 구조

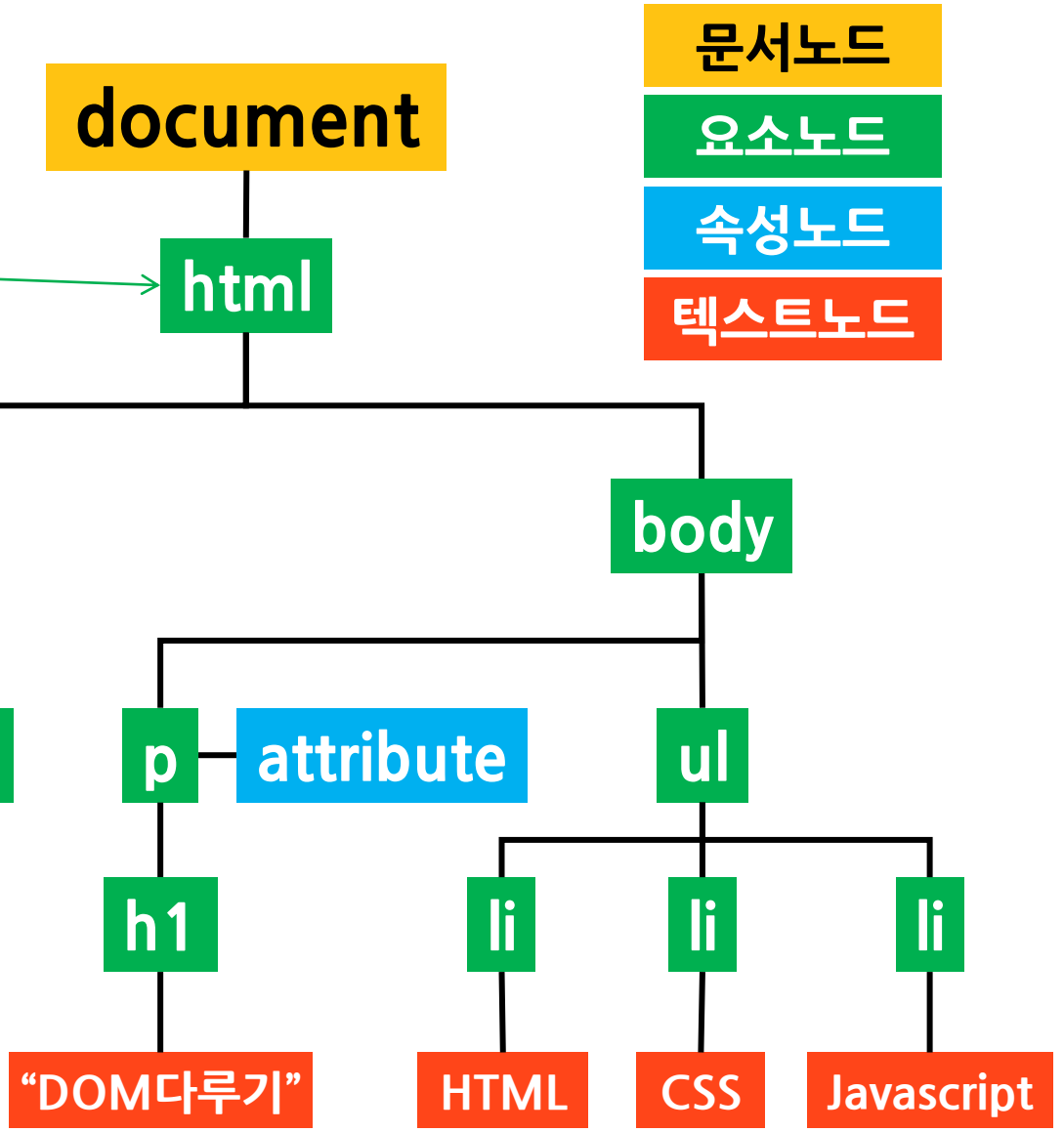
HTML문서 읽는 순서

사용자가 웹 페이지 방문



DOM구조

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>DOM구조</title>
</head>
<body>
    <div>
        <p align="center">
            <h1>DOM다루기</h1>
        </p>
        <ul>
            <li>HTML</li>
            <li>CSS</li>
            <li>Javascript</li>
        </ul>
    </div>
</body>
</html>
```



HTMLElement

모든 종류의 HTML요소를 나타내는 인터페이스

* getElement* 메소드를 통해서 원하는 객체를 조회
조회된 객체들을 대상으로 구체적인 작업 처리

Element 요소 접근

HTMLElement객체 반환

함수	설명
getElementById(id)	특정 아이디를 가진 요소 조회
getElementsByName(name)	Name속성을 가진 요소 조회
getElementsByTagName(tagname)	태그 이름을 기준으로 요소 조회
getElementsByClassName(class)	특정 클래스를 가진 요소 조회

HTMLCollection객체 반환

Element 요소 접근 메소드

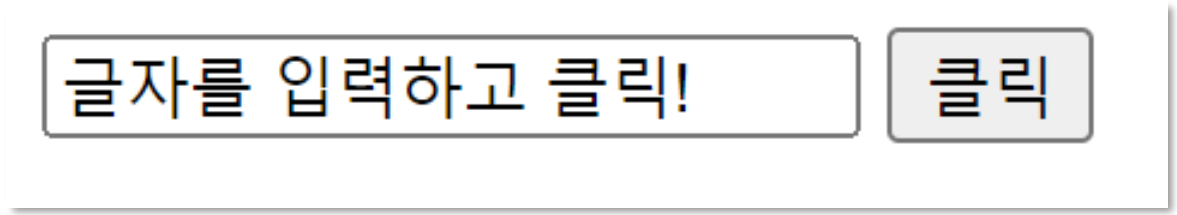
querySelector(selector)

< CSS 선택자를 이용하여 하나의 요소를 검색 >

querySelectorAll(selector)

< CSS 선택자를 이용하여 모든 요소를 검색 >

getElementById() 실습



ex10dom_js.html

접근하고자 하는 요소의 id 입력

getElementById()

< 접근하고자 하는 HTML태그의 id값을
이용해 HTMLElement객체 조회 >

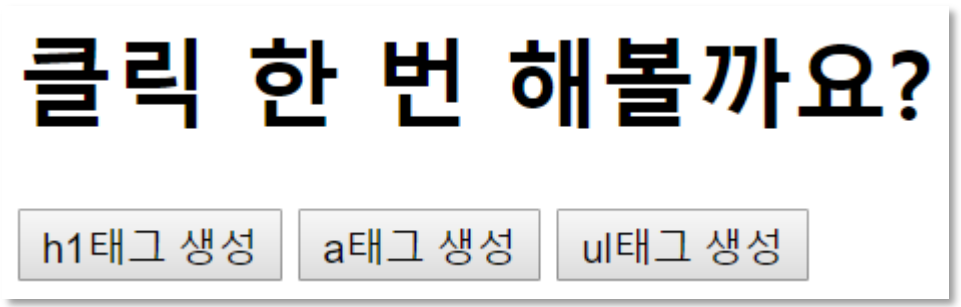
getElementById()

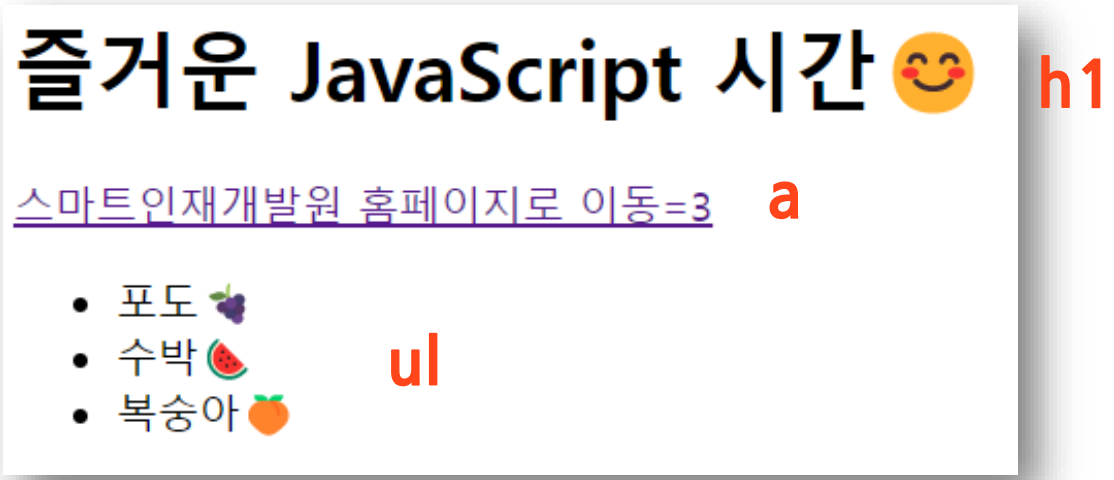
```
<body>
        <p id="text">Hello JavaScript</p>
        <button onclick="innerFunc()">클릭</button>
<script>
function innerFunc(){
        var str = document.getElementById("text");
        alert(str.innerHTML);
}
</script>
</body>
```

태그 내 콘텐츠(내용)
읽어오거나
저장하기 위한 속성

getElementById() 실습

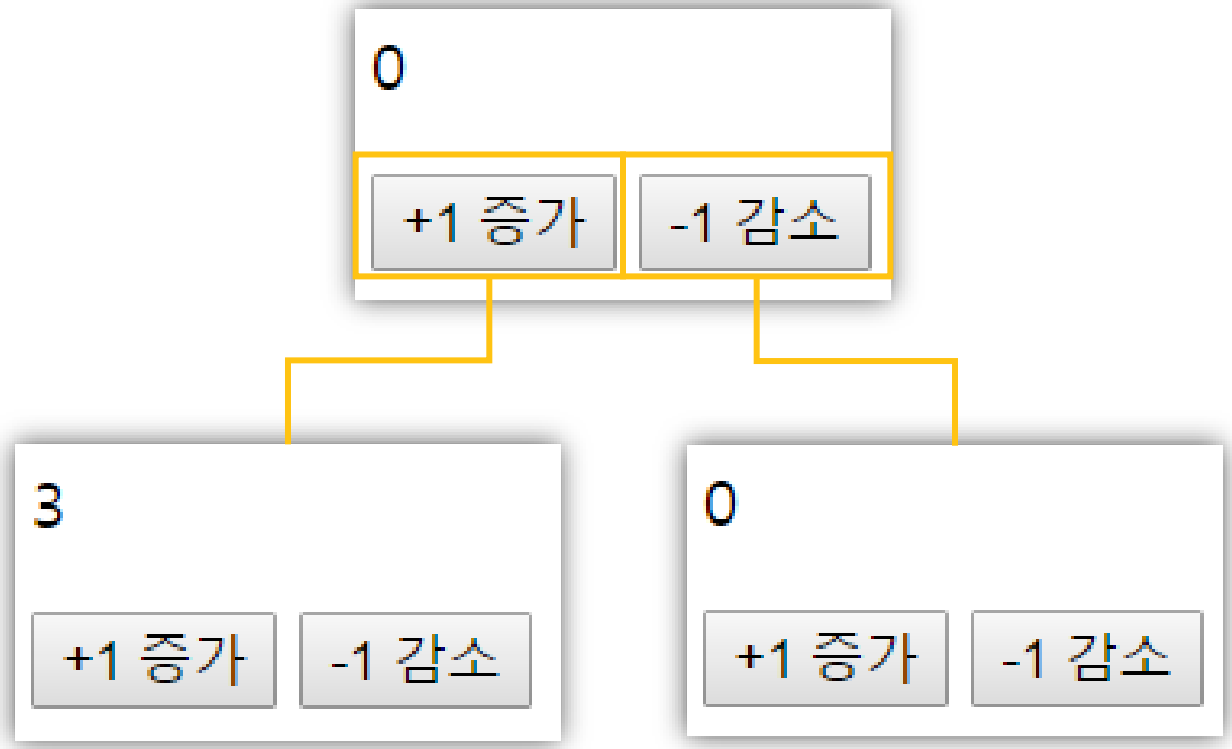
태그를 생성하는 버튼을 클릭했을 때,
다음과 같은 내용이 출력되는 페이지를 만드시오.





getElementById() 실습

버튼 클릭 시, 1씩 증감되는 웹 페이지를 만드시오.
(단! 0일 경우 감소되지 않도록!)



ex09dom_js.html

.innerHTML vs .value

HTMLElement.innerHTML

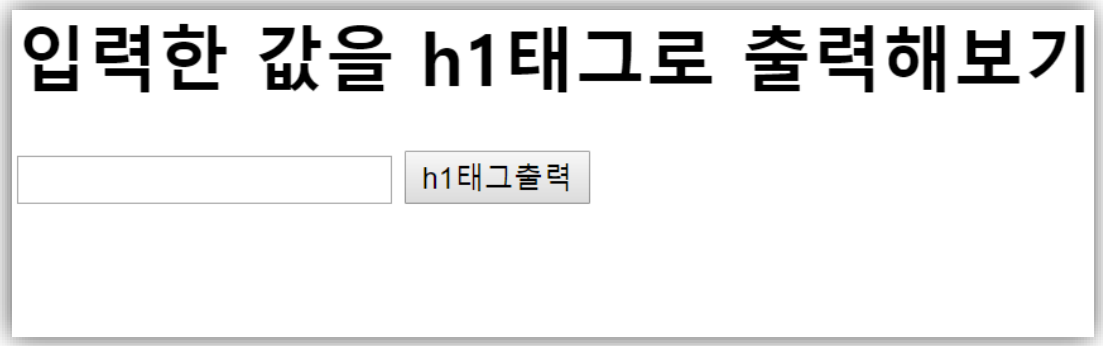
< 태그 내에 새로운 요소객체나 내용을 저장하거나 가져올 때 사용 >

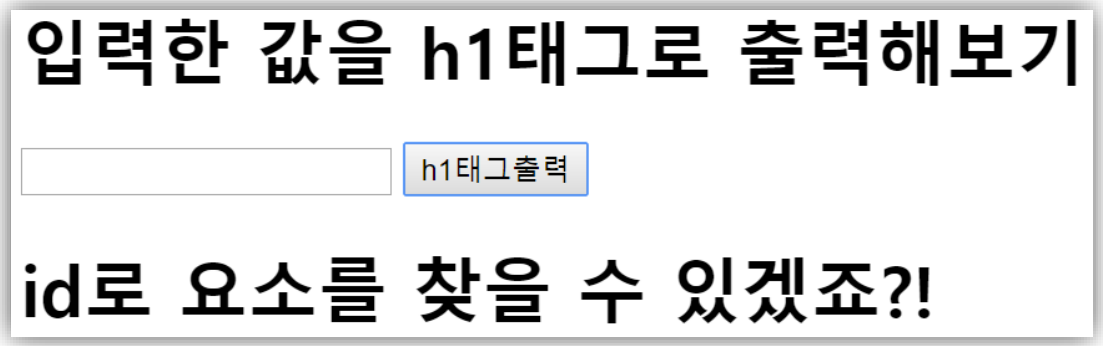
HTMLInputElement.value

< input 태그의 value값을 저장하거나 가져올 때 사용 >

getElementById() 실습

버튼 클릭 시 , 입력한 값을 div태그 내 h1 태그로 출력되는 웹 페이지를 만드시오.





ex11dom_js.html

접근하고하 하는 요소의 이름 입력

getElementsByTagName()

접근하고자 하는 HTML태그의 이름을 이용하여
HTMLCollection객체 조회

getElementsByTagName() 실습

```
<body>
<ul>
        <li>HTML</li>
        <li>CSS</li>
        <li>JavaScript</li>
</ul>
<script>
        var lis = document.getElementsByTagName("li");
        for(var i=0; i<lis.length; i++){
                alert(lis[i].innerHTML);
        }
</script>
</body
```

ex12dom_js.html

Element 요소 변경 메소드

메소드	설명
Element.innerHTML = new html content	내부 내용 변경 ex) div.innerHTML='hello'
Element.attribute = new value	속성 변경 ex) img.src='a.png'
Element.setAttribute(attribute,value)	속성 변경 ex) img.setAttribute('src','a.png')
Element.style.property	스타일 변경 ex) div.style.color='red'

Element 스타일 변경

HTML요소객체.style.property = 속성값

ex) h1태그의 글자색을 초록색으로 변경

```
<h1 id="h1Tag">h1태그</h1>
<button onclick="inputFunc()">클릭</button>

<script>
    document.getElementById("h1Tag").[redacted]="green";
</script>
```

ex14dom_js.html

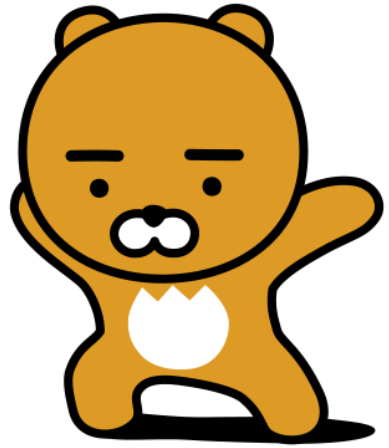
자바스크립트에서의 this

- 자기 참조 변수(Self-reference variable)이다.
- **함수** 실행시에는 **전역(window)객체** 가리킴
- **생성자 내부 함수(메소드)** 실행시에는 **메소드를 소유하고 있는 객체**를 가리킴
- **생성자** 실행시에는 **새롭게 만들어진 객체**를 가리킴

Element 속성 변경

이미지 변경

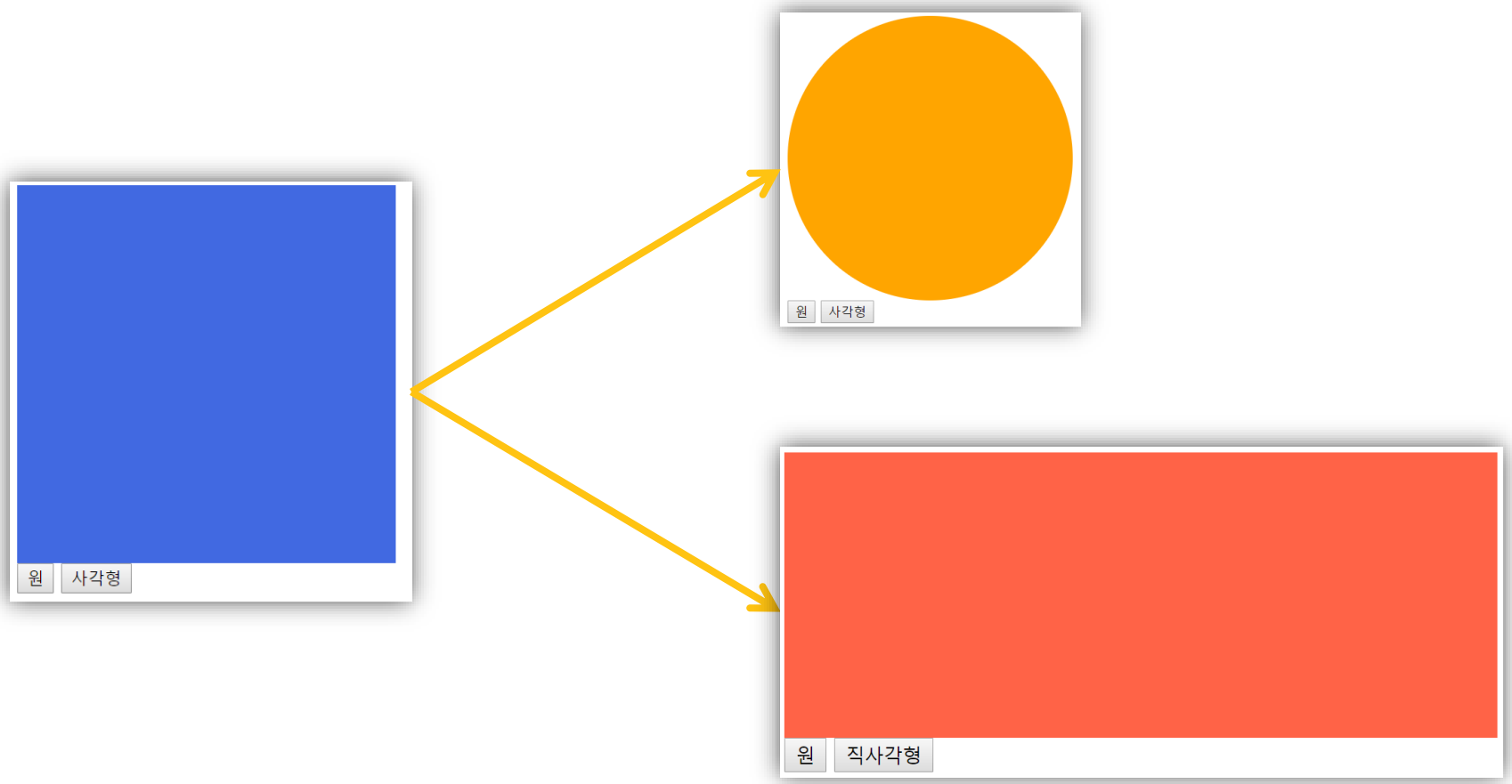
click



이미지 변경

ex01dom_js.html

Element 스타일 변경 실습

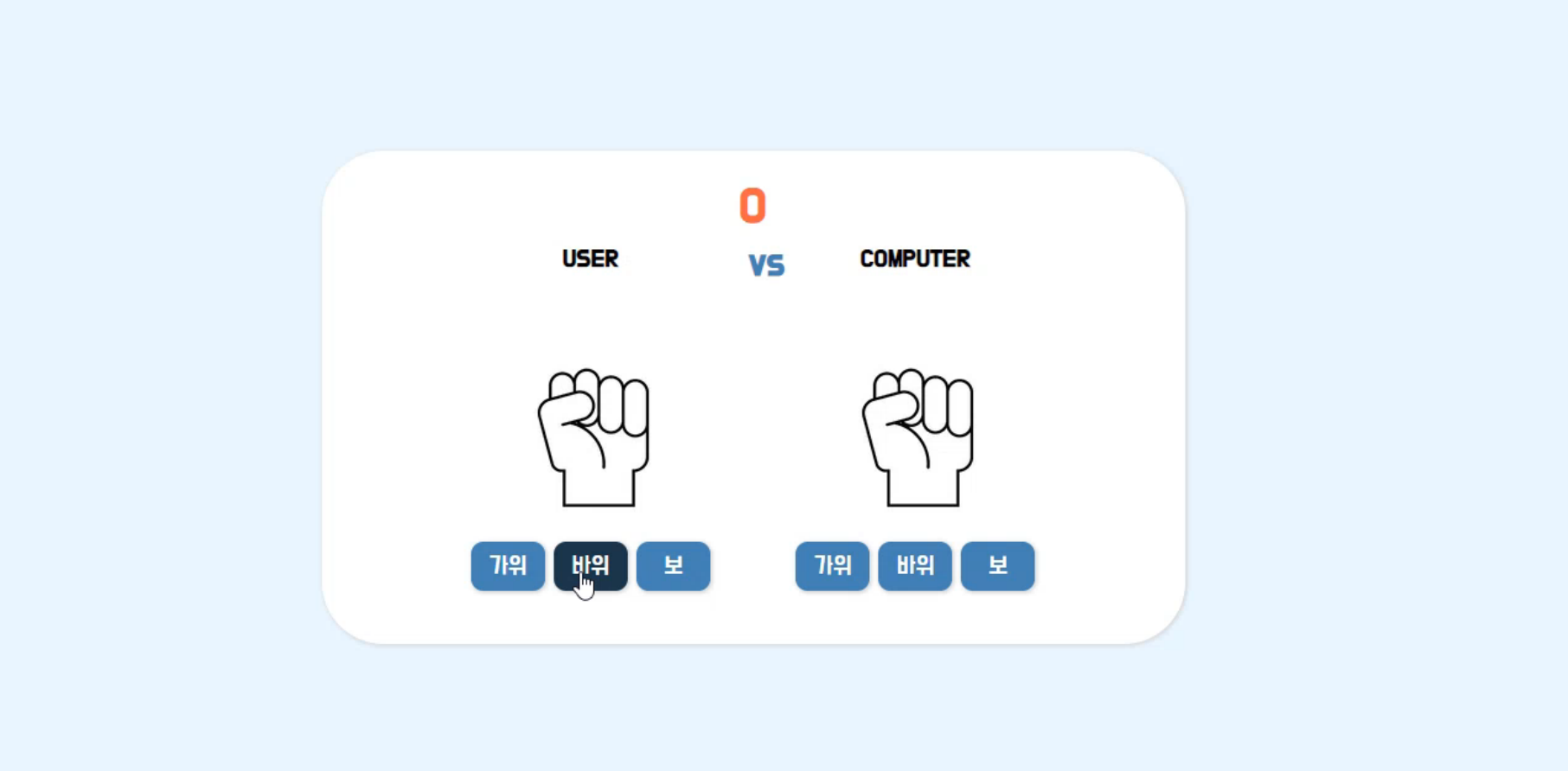


ex02dom_js.html

Element 시계 만들기 실습

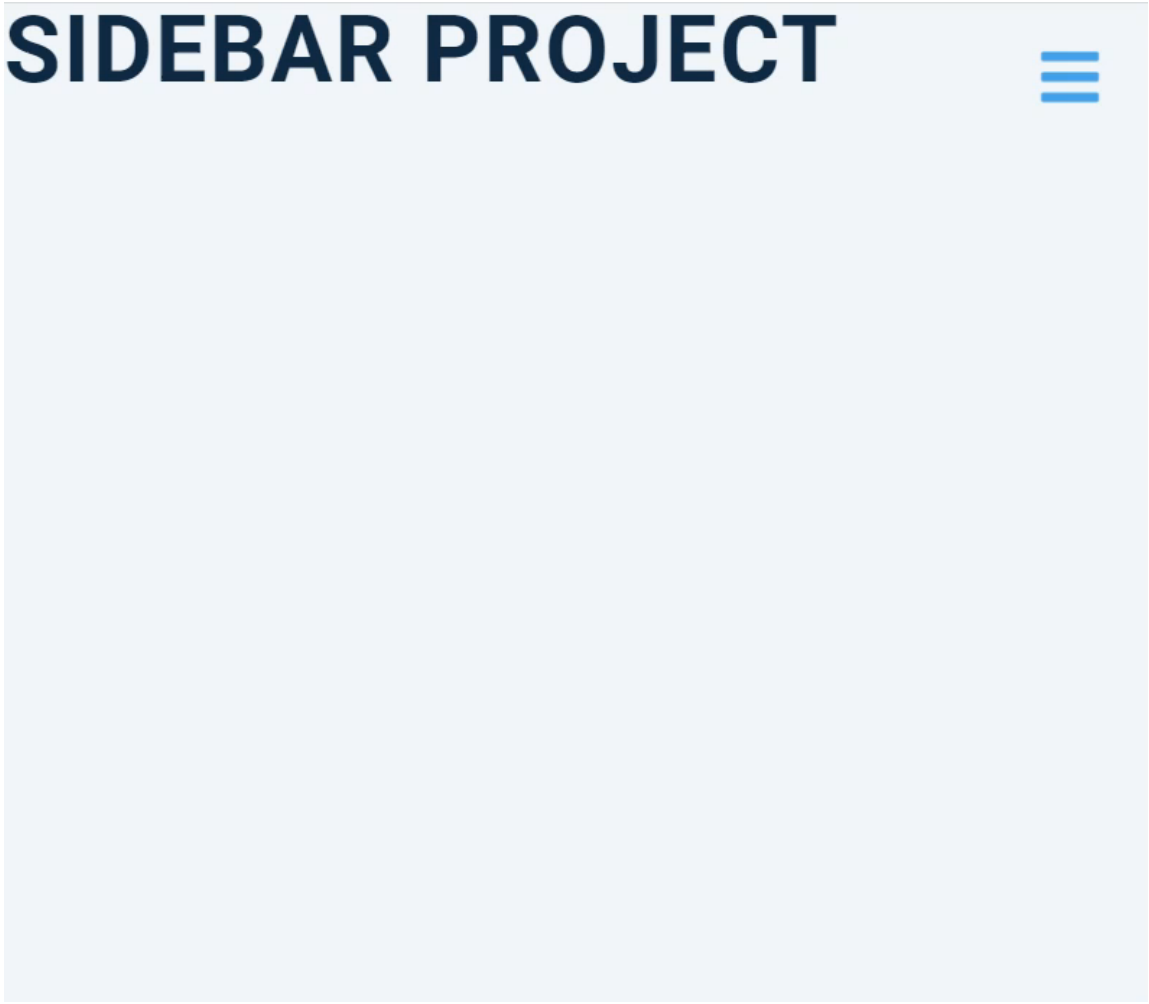
ex03clock.html

Element 시계 만들기 실습



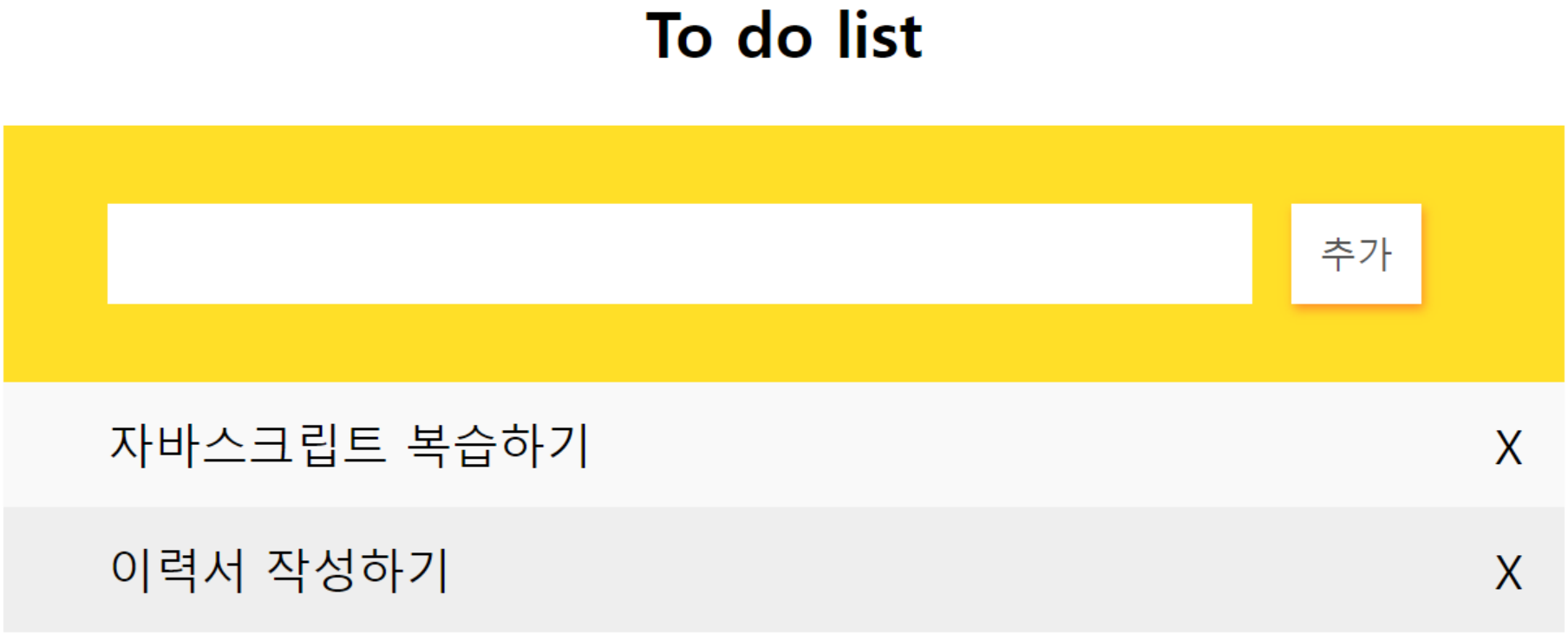
Ex04rock.html

Element 시계 만들기 실습



ex04sidebar.html

Element 속성 변경



Element 속성 변경

Element 속성 변경

1) 필요한 태그들 가져오기

2) 할 일을 하나로 묶어줄 배열 생성하기

3) 버튼을 클릭했을 때 input태그 내부 값 가져와서 배열에 넣기

4) 가져온 데이터를 보여주기(ul 태그 형식으로 div 안에 글자)

5) X버튼 클릭 시 이벤트 추가하기

{ 다음시간에 배울 내용 }
- 요소조작, 이벤트