

Abstract geometric lines in the top left corner, consisting of several overlapping, irregular polygons and lines in a light beige color, creating a layered, architectural feel.

WEB프로젝트

JAVASCRIPT

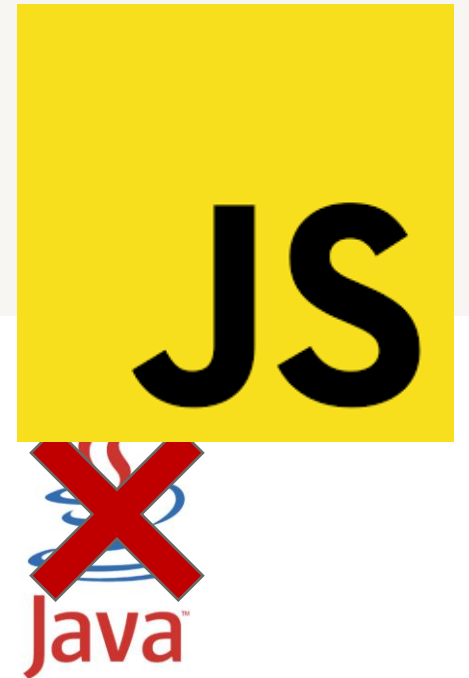
HTML, CSS와 함께 웹을 구성하는 대표적인 요소 중 하나다.


HTML – 웹페이지의 기본 구조

CSS – 웹페이지의 디자인


JS – 웹페이지의 동적인 기능

Javascript는 동적인 웹페이지를 만들기 위해 사용되는 언어다.
클라이언트 측에서 실행되는 스크립트 언어로, 사용자와의 상호작용, 이벤트,
데이터, 비동기 통신 등을 처리하는 데 사용된다.





자바스크립트의 특징

1. 인터프리터 언어
 2. 객체 지향 프로그래밍
 3. 이벤트 기반 프로그래밍
 4. 동적 타입 언어
 5. 클라이언트, 서버 개발
- 

1. 인터프리터 언어

인터프리터란?

코드를 한 줄씩 읽어 내려가며 실행하는 프로그램 (자바스크립트, 파이썬, R, SQL 등)

자바스크립트는 인터프리터 언어라서 별도의 컴파일 과정이 필요하지 않으며,
소스 코드를 한 줄 단위로 즉시 해석하고 실행한다.

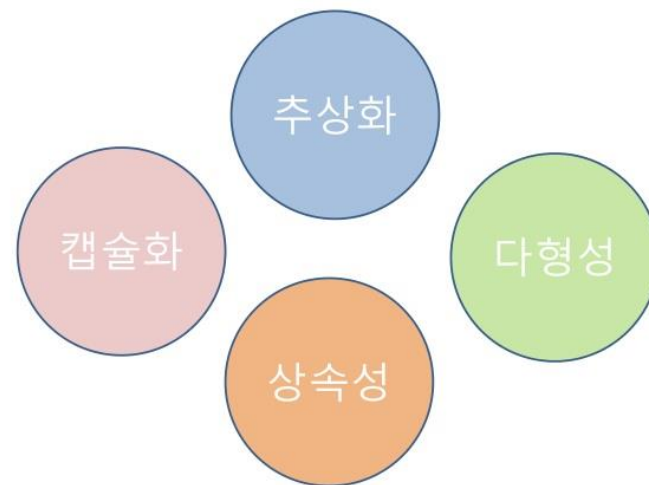
인터프리터의 반대 = 컴파일러

2. 객체 지향 프로그래밍

자바스크립트는 객체 지향 프로그래밍의 개념을 지원하며,
JAVA, C# 같은 객체 지향 언어와 유사한 방식으로 개발이 가능하다.

속성과 메소드를 가지는 객체를 생성 가능.
상속, 캡슐화, 추상화, 다형성의 객체 지향 프로그래밍의 특징도
동일하게 적용된다.

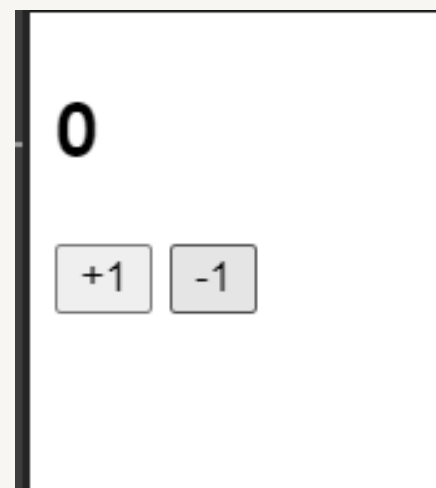
객체지향의 4대 특징



3. 이벤트 기반 프로그래밍

자바스크립트는 이벤트 기반 프로그래밍을 지원한다.

웹페이지, 웹어플리케이션에서 발생하는 다양한 이벤트 (클릭, 마우스 오버, 키보드 입력, 페이지 로딩, 스크롤 등)에 대한 응답으로 코드를 실행하여 이벤트에 따른 동작을 구현할 수 있다.

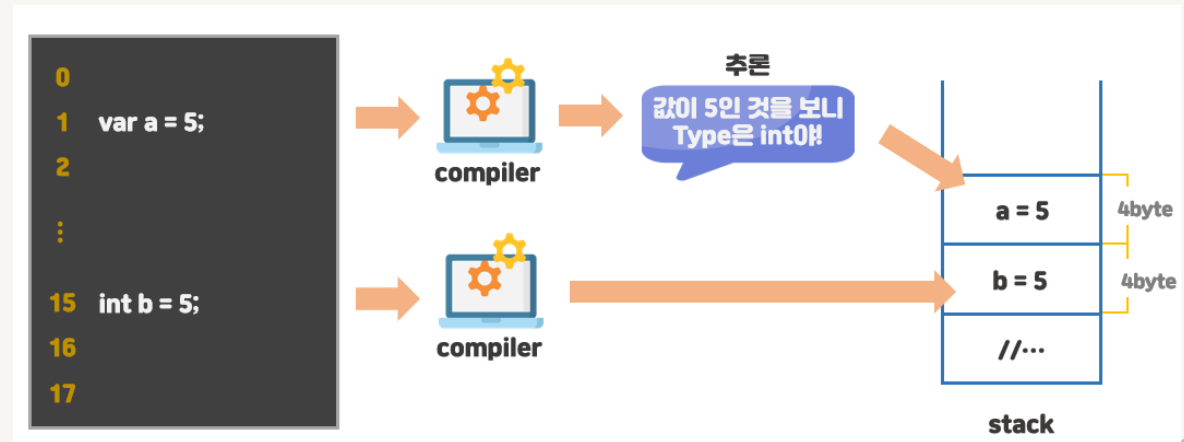


4. 동적 타입

자바스크립트는 동적 타입 언어라는 특성을 가지고 있다.

변수 선언 시, (**int a = 1, double b = 1.5**)와 같은 식으로
변수의 자료형을 지정해야 되는 정적 타입 언어(C, Java)와 반대로

(**var a = 1, var b = 1.5**) 와 같이 모든 변수를 var를 붙여서 선언하면
데이터의 타입을 자동으로 추론하여 할당한다.
(var 대신 let, const를 사용하기도 한다)



5. 클라이언트, 서버 개발

자바스크립트는 기본적으로 클라이언트 개발에 사용되는 언어다.

기존에는 클라이언트를 개발하려면 HTML, CSS, Javascript를 사용하고 서버를 개발하기 위해서는 Java등의 다른 언어를 사용했다.
하지만 node.js를 통해 Javascript 언어를 통해 백엔드 서버 개발이 가능해졌다.

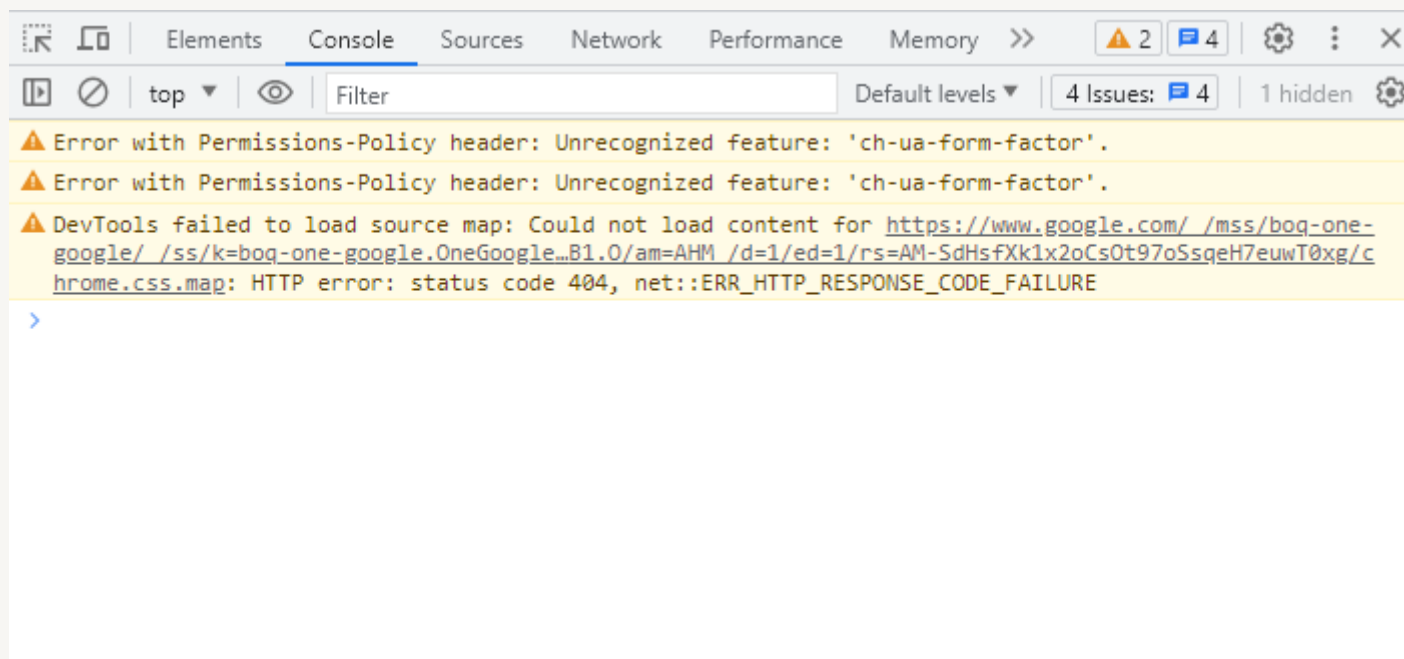
따라서 Javascript 언어를 학습하면 프론트엔드, 백엔드 개발하는데 모두 사용할 수 있다.



자바스크립트 실행

자바스크립트의 개발환경은 HTML, CSS와 마찬가지로 Visual studio code를 사용하지만, 웹브라우저의 콘솔을 통해 자바스크립트 코드를 실행하는 것도 가능하다.

F12 > Console



브라우저 콘솔에 다음과 같이

name 변수와 age 변수를 각각 선언하고

console.log 함수와 alert함수로 어떤 값이 출력되는지 확인한다.

console.log 함수로 콘솔창에 name이 출력되고
alert 함수로 입력한 나이가 브라우저 알림창에 출력이 된다.

name 변수가 2번 선언 되어, 처음 name에 들어간 "Lee" 라는 값이
Kim으로 덮어쓰여진다.



```
var name = "Lee";  
var age = 50;  
var name = "Kim";  
  
console.log(name);  
  
alert(age);
```

VAR와 LET CONST

var로 변수를 선언할 경우, 중복 선언을 허용하기 때문에
이미 선언한 변수를 뒤에서 다시 선언하여 값이 의도치 않게 변하는 상황이 발생할 수 있다.

이런 문제를 방지하기 위해서는 자바스크립트 ES6 업데이트에서 추가된
let, const를 사용하면 된다.

let – 같은 이름을 가진 변수의 재선언은 불가능하지만 값을 재정의 하는 것은 가능하다 (변수)

const – 같은 이름을 가진 변수의 재선언, 값의 재정의 모두 불가능하다 (상수)

```
var x = 2; //변수 선언
```

```
x = 4; // 재정의
```

```
var x = 4; // 재선언
```

VAR와 LET CONST

var로 변수를 선언할 경우, 중복 선언을 허용하기 때문에
이미 선언한 변수를 뒤에서 다시 선언하여 값이 의도치 않게 변하는 상황이 발생할 수 있다.

이런 문제를 방지하기 위해서는 자바스크립트 ES6 업데이트에서 추가된
let, const를 사용하면 된다.

let – 같은 이름을 가진 변수의 재선언은 불가능하지만 값을 재정의 하는 것은 가능하다 (변수)

const – 같은 이름을 가진 변수의 재선언, 값의 재정의 모두 불가능하다 (상수)

```
var x = 2; //변수 선언
```

```
x = 4; // 재정의
```

```
var x = 4; // 재선언
```

자료형과 출력

자바스크립트는 Number, String, Boolean 등 다른 언어와 비슷한 자료형을 가지지만, Undefined처럼 다른 언어와는 다른 자료형도 존재한다.

우측의 코드를 작성하여 변수의 출력 형식과 자료형에 따른 값이 어떻게 출력되는지 확인해보자

Number	정수, 실수 등의 숫자
String	문자, 문자열
Boolean	True, False
Null	Null, 값이 비어있음, 식별되지 않은 값
Undefined	값이 할당되지 않은 경우 Undefined를 반환

```
const name = "Kim";
const message = `My name is ${name}`;
const message2 = "My name is ${name}";

console.log(message)
console.log(message2)

const a = "나는 ";
const b = "입니다.";
const age = 30;

console.log(a + age + "살" + name + b)

const n = null;
const u = undefined;

console.log(n)
console.log(u)
```

브라우저 모달 창

모달 창 : 원래 페이지로 돌아가기 전에 사용자의 상호작용을 요구하는 창 (팝업 창과 다름)

자바스크립트로 브라우저에서 제공하는 창을 이용할 수 있다.

prompt = 사용자 입력창

alert = 알림(경고)창

confirm = 확인창

장점 - 쉽고 빠르게 구현 가능하다

단점 - 해당 창이 떠 있는 동안 기존 페이지의 스크립트가 정지된다.
스타일링이 불가능하다.

chrome://new-tab-page 내용:

알림창입니다

확인

```
const name = prompt("이름을 입력해주세요.", "Kim");  
alert("안녕하세요, " + name + "님");  
//alert(`안녕하세요, ${name}님`);  
  
const del = confirm("정말 삭제하시겠습니까?");  
console.log(del);
```

형 변환

오른쪽과 같은 코드를 작성하여
사용자로부터 두개의 점수를 입력받아 평균을 출력하는
프로그램을 작성하고 결과를 확인해보자

두개의 점수를 입력하고 결과를 확인하면
전혀 다른 값이 나오는 것을 볼 수 있다.

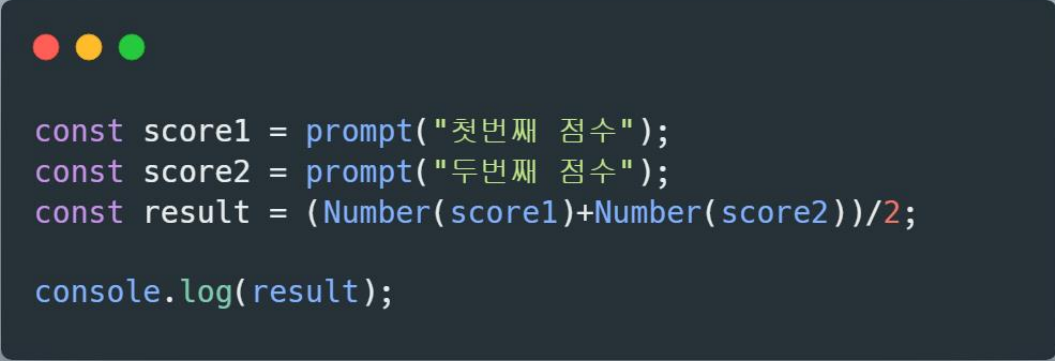
80, 90을 입력하면 170을 2로 나눈 값인 85가 나와야 되는데
4045라는 결과값이 출력이 된다.

이는 prompt로 값을 입력받은 값은 문자열로 처리 되어
발생하는 문제다.

이를 해결하기 위해서 prompt로 입력받은 값을
명시적 형변환 해줄 필요가 있다.

```
const score1 = prompt("첫번째 점수");
const score2 = prompt("두번째 점수");
const result = (score1+score2)/2;

console.log(result);
```



```
const score1 = prompt("첫번째 점수");  
const score2 = prompt("두번째 점수");  
const result = (Number(score1)+Number(score2))/2;  
  
console.log(result);
```

따라서 위와 같이 입력받은 score1, score2 값을 Number()로 감싸서 형변환을 해주면 값이 의도대로 출력이 된다.

Number 외에도 String, Boolean 형변환 또한 가능하다.

자바스크립트 함수

자바스크립트의 가장 기본적인 함수 선언 방식으로 다른 언어와 동일한 형태로 함수 선언이 가능하다.

function {함수이름(매개변수)}

또한, ES6 업데이트에 추가된 함수 선언 방식으로 화살표 함수가 있다.

function 표기를 생략하여 함수를 선언할 수 있으며, 함수 표현식이 한줄인 경우, {} 중괄호와 return도 생략한 형태로 함수를 선언할 수 있다.

```
//선언
function add(x, y) {
  let temp = x + y;
  return temp;
}

//실행
add(1, 2) // 3
```

```
//화살표 함수
add = (x, y) => {
  let temp = x + y;
  return temp;
}

//실행
add(1, 2) // 3

//화살표 함수 축약
add = (x, y) => x + y;
//실행
add(1, 2)
```

조건문

조건문은 if문을 사용하며

else if, else를 사용하여 조건을 이어나갈 수 있다.

코드 하단의

if(1000 == "1000")의 조건문은
숫자 1000과 문자 "1000"은 서로 다르기 때문에
조건문이 실행되지 않아야 되지만, "같습니다"가
출력 되는 걸 볼 수 있다.

이는 다른 언어처럼 == 비교연산자를 사용할 시
자동으로 형변환을 적용하여 비교를 한다.

따라서 형변환이 적용되지 않은 값을 비교하고 싶다면
=== 처럼 =를 3개 붙여서 식을 작성하면 된다.

```
let money = 10000

if(money > 20000) {
  console.log("오늘 저녁은 소고기");
}
else if(money > 10000) {
  console.log("오늘 저녁은 치킨");
}
else{
  console.log("오늘 저녁은 없다");
}

if(1000 == "1000"){
  console.log("같습니다");
}

if(1000 === "1000"){
  console.log("같습니다");
}
```

반복문

for 반복문을 사용할 수 있다.

```
for(let i = 0; i < 10; i++){  
  console.log("반복문", i);  
}
```

횟수를 지정하여 반복하는 for 반복문이 아닌

특정 배열의 요소만큼 반복하여 각 요소를 가져오는
forEach문도 이용할 수 있다.

```
myArray = [1,2,3,4,5]  
  
myArray.forEach(number => {  
  console.log("반복문 "+number);  
});
```

배열과 배열 객체

여러 변수를 묶어서 배열 형태로 선언도 가능하다.

또한 배열과 관련된 작업을 쉽게 할 수 있도록 도와주는 다양한 Array 메소드가 존재한다.

push - 배열 마지막에 요소를 추가한다

pop - 배열 마지막 요소를 제거하고 반환한다

shift - 배열 첫 요소를 제거하고 반환한다

unshift - 배열 가장 앞에 요소를 추가한다

reverse - 배열의 순서를 반대로 뒤집는다

sort - 배열 요소를 알파벳 순서로 정렬한다.

이외에도 join, slice, concat, toString, forEach, map 등 다양한 메소드를 지원한다.



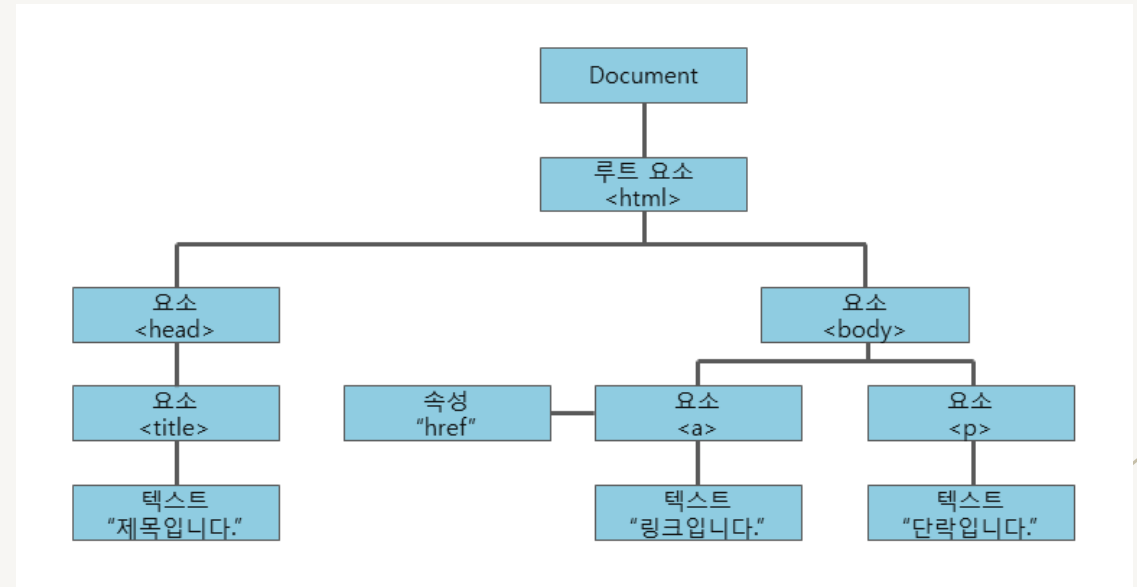
```
myArray = [1,2,3,4,5];  
  
Array.isArray(myArray);//true  
  
myArray.push(6);  
myArray.pop();  
myArray.shift();  
myArray.unshift(0);  
myArray.reverse();  
  
myArray2 = [5,1,3,2,4];  
  
myArray2.sort();
```

DOM

DOM(Document Object Model)은 HTML 문서에 접근하기 위한 인터페이스의 일종이다.
문서 내의 모든 요소를 정의하고, 각각의 요소에 접근하는 방법을 제공한다.

자바스크립트는 DOM 객체 모델을 이용하여 다음과 같은 작업을 할 수 있다.

- 새로운 HTML 요소, 속성 추가, 제거, 변경
- HTML 문서의 모든 CSS 스타일 변경
- HTML 문서에 새로운 HTML 이벤트 추가, 이벤트 반응



DOCUMENT 객체

Document 객체는 웹페이지 자체를 의미하며,
웹페이지에 존재하는 HTML 요소에 접근하려면 반드시 Document 객체를 통해 접근해야 된다.

Document 객체도 마찬가지로 HTML 요소와 관련된 작업을 도와주는 메소드를 제공한다.

- HTML 요소의 선택
- HTML 요소의 생성
- HTML 이벤트 핸들러 추가
- HTML 객체의 선택

문서 객체 메소드

Document 객체는 웹페이지 자체를 의미하며,
웹페이지에 존재하는 HTML 요소에 접근하려면 반드시 Document 객체를 통해 접근해야 된다.

Document 객체도 마찬가지로 HTML 요소와 관련된 작업을 도와주는 메소드를 제공한다.

- HTML 요소의 선택
- HTML 요소의 생성
- HTML 이벤트 핸들러 추가
- HTML 객체의 선택

DOM 요소 접근을 위해
다음과 같은 HTML 코드를 작성한다.

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>DOM</title>
</head>
<body>
  <h1>다양한 선택자</h1>
  <ul>
    <li>첫 번째 아이템이에요!</li>
    <li class="even">두 번째 아이템이에요!</li>
    <li>세 번째 아이템이에요!</li>
    <li class="even">네 번째 아이템이에요!</li>
    <li>다섯 번째 아이템이에요!</li>
  </ul>
  <p>
    <span id="blue">아이디 선택자로 요소를 선택합니다.</span><br>
    <span class="green">클래스 선택자로 요소를 선택합니다.</span><br>
  </p>
</body>
</html>
```

다양한 선택자

- 첫 번째 아이템이에요!
- 두 번째 아이템이에요!
- 세 번째 아이템이에요!
- 네 번째 아이템이에요!
- 다섯 번째 아이템이에요!

아이디 선택자로 요소를 선택합니다.
클래스 선택자로 요소를 선택합니다.



```
<script>
  var selectedItem = document.getElementsByTagName("li"); // 모든 <li> 요소를 선택함.
  for (var i = 0; i < selectedItem.length; i++) {
    selectedItem.item(i).style.color = "red"; // 선택된 모든 요소의 텍스트 색상을 변경함.
  }

  var selectedItem = document.getElementById("blue"); // blue Id를 가진 요소를 선택
  selectedItem.style.color = "blue";

  var selectedItem = document.getElementsByClassName("green"); // green 클래스를 가진 요소를 모두 선택
  selectedItem[0].style.color="green"; //

  var selectedItem = document.querySelectorAll("li.even"); // even 클래스를 가진 li태그를 모두 선택
  for (var i = 0; i < selectedItem.length; i++){
    selectedItem.item(i).style.color = "yellow";
  }
</script>
```

body 태그 안에 <script>태그를 생성하여 다음과 같이 코드를 작성하면
선택한 HTML 요소의 색상만 변경이 되는 것을 볼 수 있다.

다양한 선택자

- 첫 번째 아이템이에요!
- 두 번째 아이템이에요!
- 세 번째 아이템이에요!
- 네 번째 아이템이에요!
- 다섯 번째 아이템이에요!

아이디 선택자로 요소를 선택합니다.
클래스 선택자로 요소를 선택합니다.



```
document.write("document.write로 추가한 문구입니다.")
document.write("<p class='dom' style='text-align: center;'>태그로 입력할 수도 있습니다.</p>");

var str = document.getElementsByClassName("dom");
str[0].innerHTML = "document.write로 추가한 문장을 innerHTML로 수정했습니다.";
```

<script> 태그 안에 위의 코드를 작성한다.

기존의 있던 내용의 수정 뿐 아니라 document.write 메소드를 사용해 새로운 요소를 추가하고, 텍스트 뿐 아니라 요소의 클래스, 스타일도 지정할 수 있다.

다양한 선택자

- 첫 번째 아이템이에요!
- 두 번째 아이템이에요!
- 세 번째 아이템이에요!
- 네 번째 아이템이에요!
- 다섯 번째 아이템이에요!

아이디 선택자로 요소를 선택합니다.
클래스 선택자로 요소를 선택합니다.

document.write로 추가한 문구입니다.

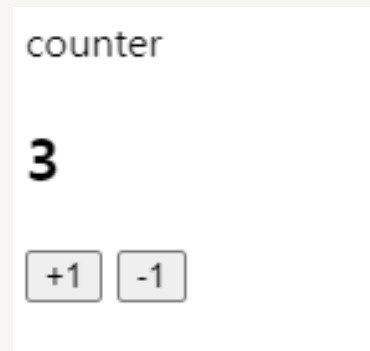
document.write로 추가한 문장을 innerHTML로 수정했습니다.

간단한 이벤트 처리

간단한 자바스크립트 이벤트 처리를 통해
+1, -1 버튼을 누르면 숫자가 증가, 감소하는
숫자 카운터를 구현한다.

counter 폴더를 만들고
index.html
index.js

파일을 생성한다.



index.html

```

<!DOCTYPE html>
<html>
<head>
  <title>숫자 카운터</title>
  <meta charset="UTF-8" />
</head>
<body>
  <div class="container">
    <header>
      counter
    </header>
    <h2 id="number">0</h2>
    <div class="btn">
      <button id="increase">+1</button>
      <button id="decrease">-1</button>
    </div>
  </div>
  <script src="index.js"></script>
</body>
</html>

```

index.js

```
const number = document.getElementById("number");
const increase = document.getElementById("increase");
const decrease = document.getElementById("decrease");

increase.onclick = () => {
  const current = parseInt(number.innerText, 10);
  number.innerText = current + 1;
};

decrease.onclick = () => {
  const current = parseInt(number.innerText, 10);
  number.innerText = current - 1;
};
```

Id 선택자로 숫자를 출력해주는 요소와 증가, 감소 버튼을 선택한다

그 후 각각의 버튼에 onclick 이벤트가 발생한 경우, 화살표 함수로 지정한 동작을 수행한다.

number의 innerText는 Text 형식이기 때문에 해당 요소를 parseInt로 정수로 변환시킨다.

변환 시킨 값을 current 상수에 저장하고, number 요소의 텍스트를 current에 1을 더한 값으로 대체한다.

응용해보기

1씩 증가, 감소 하는 카운터를 완성했다면,

이번에는 5씩 증가, 감소 하는 버튼,
누르면 초기화 되는 Clear 버튼
(그 외 자유롭게 추가 가능)
등의 요소를 추가하여 실제로 작동 되도록
HTML 및 JS 파일을 수정해보자.

또한, 실제 다른 사람들이 이용하는 서비스라 생각을 하고
스타일을 적용하여 디자인 요소를 추가한다

디자인 보조 사이트

<https://www.figma.com/>



이벤트 처리로 CSS 수정

prompt, alert 같은 브라우저 자체 기능이 아닌
자바스크립트로 직접 구현한 간단한 모달 창을 만들어본다.

해당 모달 창은 기본적으로 display: none 속성이 적용되어
사용자에게 보이지 않는 상태로 존재한다.

사용자가 열기 버튼을 클릭하는 이벤트가 발생하면
자바스크립트에서 display 속성을 수정하여
사용자에게 숨겨져 있던 모달 창을 보여주는 기능이다.

이를 통해 자바스크립트로 HTML의 요소 뿐만 아니라
요소에 적용되는 스타일도 수정하는 방법을 배울 수 있다.

모달 창 TEST

열기 버튼을 누르면 숨겨져 있던 모달 창이 나타납니다

열기

안녕하세요

자바스크립트로 display속성을 수정하여 모
달 창을 나타냅니다

닫기

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Modal</title>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="style.css">
  </head>

  <body>
    <h1>모달 창 TEST</h1>
    <p>열기 버튼을 누르면 숨겨져 있던 모달 창이 나타납니다</p>
    <button id="open">열기</button>
    <div class="modal-wrapper" style="display: none;">
      <div class="modal">
        <div class="modal-title">안녕하세요</div>
        <p>자바스크립트로 display속성을 수정하여 모달 창을 나타냅니다</p>
        <div class="close-wrapper">
          <button id="close">닫기</button>
        </div>
      </div>
    </div>
    <script src="index.js"></script>
  </body>
</html>
```


index.js

```
const open = document.getElementById("open");
const close = document.getElementById("close");
const modal = document.querySelector(".modal-wrapper");

open.onclick = () => {
  modal.style.display = "flex";
};
close.onclick = () => {
  modal.style.display = "none";
};
```

숫자 카운터와 동일하게 Id 선택자 or 쿼리셀렉터를 이용하여 HTML 요소를 가져온다.

마찬가지로 동일하게 해당 요소에 onclick 이벤트가 발생할 시 modal 요소의 display 속성을 none 에서 flex로 변경하여 사용자에게 모달 창을 보여준다.

그 후 닫기 버튼을 누르면 다시 display 속성을 none으로 변경하여 창을 숨긴다.

이미지 슬라이더

addEventListener와 자연스러운 이미지 전환을 위한
애니메이션이 적용된 이미지 슬라이더 페이지를 구현한다.



Next

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>이미지 슬라이드</title>
  <link rel="stylesheet" href="styles.css" />
</head>

<body>
  <div class="slider-container">
    <div class="slide">
      
      <h1>1</h1>
    </div>
    <div class="slide">
      <h1>2</h1>
    </div>
    <div class="slide">
      <h1>3</h1>
    </div>
    <div class="slide">
      <h1>4</h1>
    </div>
  </div>
  <div class="btn-container">
    <button type="button" class="prevBtn">
      prev
    </button>
    <button type="button" class="nextBtn">
      next
    </button>
  </div>
  <script src="index.js"></script>
</body>
</html>
```

style.css

```
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

img{
  width: 100%;
  display: block;
}

.section {
  padding: 5rem 0;
}

.section-center {
  width: 90vw;
  margin: 0 auto;
  max-width: 1170px;
}

main {
  min-height: 100vh;
  display: grid;
  place-items: center;
}

.slider-container {
  border: 5px solid #ffffff;
  width: 80vw;
  margin: 0 auto;
  height: 40vh;
  max-width: 80rem;
  position: relative;
  border-radius: 10px;
  overflow: hidden;
  margin-top: 4rem;
}
```

```
.slide {
  position: absolute;
  width: 100%;
  height: 100%;
  background: #cccccc;
  display: grid;
  place-items: center;
  transition: all 0.25s ease-in-out;
  text-align: center;
}

.slide-img {
  height: 100%;
  object-fit: cover;
}

.slide h1 {
  font-size: 5rem;
}

.person-img {
  border-radius: 50%;
  width: 6rem;
  height: 6rem;
  margin: 0 auto;
  margin-bottom: 1rem;
}

.slide:nth-child(1) h1 {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}

.slide:nth-child(2) h1 {
  color: #dddddd;
}
```

style.css

```
.slide:nth-child(2) {
  background: linear-gradient(rgba(0, 0, 0, 0.4), rgba(0, 0, 0, 0.4)),
  url("../img/img2.jpg") center/cover no-repeat;
}

.slide:nth-child(3) {
  background: url("../img/img3.jpg") center/cover no-repeat;
}

.slide:nth-child(4) {
  background: url("../img/img4.jpg") center/cover no-repeat;
}

.btn-container {
  display: flex;
  justify-content: center;
  margin-top: 0.75rem;
}

.prevBtn,
.nextBtn {
  background: transparent;
  border-color: transparent;
  font-size: 1.75rem;
  cursor: pointer;
  margin: 0 0.25rem;
  text-transform: capitalize;
  letter-spacing: 2px;
  color: #1f1f1f;
  transition: all 0.3s linear;
}

.prevBtn:hover,
.nextBtn:hover {
  color: rgb(216, 216, 216);
}
```

index.js

```
const slides = document.querySelectorAll(".slide");
const nextBtn = document.querySelector(".nextBtn");
const prevBtn = document.querySelector(".prevBtn");
slides.forEach(function (slide, index) {
  slide.style.left = `${index * 100}%`;
});
let counter = 0;
nextBtn.addEventListener("click", function () {
  counter++;
  imgslide();
});

prevBtn.addEventListener("click", function () {
  counter--;
  imgslide();
});

function imgslide() {
  if (counter < slides.length - 1) {
    nextBtn.style.display = "block";
  } else {
    nextBtn.style.display = "none";
  }
  if (counter > 0) {
    prevBtn.style.display = "block";
  } else {
    prevBtn.style.display = "none";
  }
  slides.forEach(function (slide) {
    slide.style.transform = `translateX(-${counter * 100}%)`;
  });
}

prevBtn.style.display = "none";
```