

비전공자 웹프로그래밍을 위한

javascript문법책

발행일 : 2021.01.01

변경일 : 2021.09.23 (2본 시작일)

저자 : 박수민

이메일 : foxman12@hanmail.net

발행처 : 휴먼교육

html -

목차

03. javascript	3
> 01. javascript 기초	3
> 02. javascript 파일의 위치	5
> 03. 상수 변수 그리고 자료형	7
> 04. 다양한 연산자	13
> 05. 내장함수	17
> 06. 조건문	20
> 07. 반복문	24
> 08. 함수	27
> 09. 문자열로 구구단 구현	31
> 10. Timer 사용하기	33
> 11. 배열과 배열관련 메소드	38
> 12. 자바스크립트 객체 json	50
> 13. 이벤트	53
> 14. DOM을 이용한 이벤트	56
> 14. dom에서 내용 읽어오기	63
> 16. addEventListener 사용한 이벤트 처리	66
> 17. 자판기 만들기	68
> 18. 속성 변경 및 페이지 이동	69
> 19. 시간 계산	71
> 20. 배열 예제 1	82
> 21. 배열 예제 2	84
> 22. DOM을 이용한 CSS 변경	87
> 23. javascript ClassList 조작	92
> 24. button을 사용한 active 표현하기	95
> 26. 변수의 scope	97
> 27. 슬라이더 만들기	99
> 28. DOM을 이용한 엘리먼트 생성	104
> 32. DOM 조작	105
> 33. 정규식	108
> 31. 구글차트 사용하기	113
> 29. 미로찾기 프로젝트 1	114
> 30. 미로찾기 프로젝트 2 (리플레이)	118
> 31. 미로찾기 프로젝트 3 (최단거리)	123
> 32. 미로찾기 프로젝트 4 (자동길찾기)	129

03. javascript

> 01. javascript 기초

JavaScript는 자바와 이름만 비슷하며, 완전히 다른 프로그래밍 언어입니다. JavaScript는 웹 브라우저에서 동작하는 프로그래밍 언어로 HTML과 CSS와 함께 웹 개발의 핵심 요소 중 하나입니다. HTML은 웹 페이지의 구조와 내용을 정의하고, CSS는 웹 페이지의 스타일을 정의하고, JavaScript는 웹 페이지의 동적인 페이지 제작을 담당하여 html과 css를 조작한다. 이 세 가지 요소를 조합하여 다양한 웹 애플리케이션을 개발할 수 있습니다. web의 3요소는 html, css, javascript 이다.

표현식(Expressions)은 프로그래밍에서 의미를 가지는 가장 작은 코드 단위를 나타냅니다. 표현식은 값, 변수, 연산자 및 함수 호출의 조합으로 구성됩니다. 간단한 예제로 설명하면:

11은 숫자 표현식이며, 값 11을 나타냅니다.

+는 덧셈 연산자를 나타내며, 두 개의 피연산자를 더하는 연산을 표현합니다.

'hi'는 문자열 표현식이며, 문자열 "hi"를 나타냅니다.

var는 변수를 선언하는 표현식입니다.

if는 조건문을 시작하는 표현식입니다.

표현식을 모아서 하나의 문장(Statement)을 만들 수 있으며, 이러한 문장은 프로그램의 명령어가 됩니다. 예를 들어, 다음과 같은 두 가지 문장을 보겠습니다:

var a = 10; - 이 문장은 변수 a에 값 10을 할당하는 선언문입니다.

console.log('hello world'); - 이 문장은 콘솔에 "hello world"를 출력하는 함수 호출 문장입니다.

이러한 표현식과 문장을 조합하여 원하는 작업을 수행하는 JavaScript 프로그램을 작성할 수 있다.

키워드(Keywords): 키워드는 프로그래밍 언어에서 특별한 의미가 부여된 단어로, 변수, 함수, 제어 구조 등을 정의하거나 조작하는 데 사용됩니다. JavaScript의 키워드에는 var, if, else, for, function 등이 있습니다.

예약어(Reserved Words): 예약어는 현재는 사용되지 않지만 나중에 키워드로 예약된 단어를 의미합니다.

<script> 태그 사용: JavaScript 코드를 HTML 문서에서 실행하려면 <script> 태그안에 기술한다.

세미콜론 (;) 사용: JavaScript에서는 일반적으로 문장의 끝에 세미콜론을 붙입니다. 이것은 문법적으로 필수는 아니지만, 코드의 가독성을 높이고 오류를 방지하는 데 도움이 됩니다. 따라서 생략 가능하지만 세미콜론을 붙이는 것이 권장됩니다.

다음 코드설명과 코드를 확인해 보자.

주석(Comments): 주석은 코드에 대한 설명이나 메모를 추가할 때 사용됩니다. JavaScript에서는 한 줄 주석(//)과 여러 줄 주석(/* */) 두 가지 주석 형식을 지원합니다. 주석은 코드 실행에 영향을 미치지 않으며, 주로 코드를 이해하기 쉽게 만들거나 다른 개발자와 코드를 공유할 때 유용합니다.

문자열 출력 방법:

document.write("hello world"); 이 방법은 웹 페이지에 직접 텍스트를 쓰는 방법 중 하나입니다. 그러나 document.write는 페이지 로딩이 완료된 후에 호출되면 화면이 지워지는 문제가 발생할 수 있으므로 주의가 필요하다. 주로 테스트 목적으로만 사용됩니다.

alert("hello world"); 이 방법은 브라우저에서 경고 창을 통해 텍스트를 표시하는 방법입니다. 주로 사용자에게 알림을 표시할 때 사용됩니다. 되도록 사용하지 말자.

console.log("hello world"); 이 방법은 브라우저의 개발자 도구(DevTools)의 콘솔창에서 디버깅 정보를 출력하는 데 사용된다. 디버그 목적으로 사용되며, 사용자에게 직접적으로 출력되지 않습니다. F12를 눌러 개발자 도구의 "Console" 탭에서 결과를 확인할 수 있습니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    /* 이코드는 2023년 처음 제작한
       코드이다.*/
    document.write("hello world1<br>");
    document.write("hello world2");
    //alert은 다음에 오는 문자열이 팝업창으로 생성된다.
    alert("hello world3");
    //console을 확인하려면 f12를 누르고 디버그의 콘솔 창을 선택해서 확인해야 한다.
    console.log("hello world4");
    alert("hello world5");
  </script>
</head>
<body>
</body>
</html>
```

> 02. javascript 파일의 위치

javascript 작성위치

1. 내장형 head 안에 script태그를 기술하고 사용한다.
2. 인라인형 html태그 속성의 값으로 기술할때 사용한다.
3. 외장형 alert.js 파일을 외부에 만들고 외부에서 자바스크립트를 기술한다.

내장형 (Internal): <script> 태그를 사용하여 HTML 문서 안에 직접 JavaScript 코드를 작성하는 방식입니다. 이 코드는 페이지 로드 시 실행됩니다. 주로 <head> 태그 안에서 사용됩니다.

```
<head>

  <script>

    alert("내장형");          // JavaScript 코드 작성

  </script>

</head>
```

인라인형 (Inline): HTML 태그의 이벤트 속성 값으로 JavaScript 코드를 작성하여 해당 이벤트가 발생할 때 실행되는 방식입니다. 예를 들어, 버튼 클릭 이벤트를 처리할 때 사용됩니다.

```
<button onclick="alert('인라인형');">클릭</button>
```

외장형 (External): JavaScript 코드를 별도의 외부 파일로 저장하고, HTML 문서에서 <script> 태그의 src 속성을 사용하여 외부 파일을 연결하는 방식입니다. 외부 파일은 .js 확장자를 가집니다.

<script src="alert.js"> </script>를 이용해서 외부에서 만든 alert.js파일을 html파일에서 연결 한다.

그리고 alert.js 파일 내용:

```
alert('외장형1');
```

```
alert('외장형2');
```

가장 권장하는 방법은 외장형인데 편이상 책에서는 내장형을 사용한다. 정확한 사용방법은 아래 코드를 확인해 보자.

```
<!DOCTYPE html>
```

```
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script src="alert.js"></script>//해당 파일안의 자바스크립트 코드가 실행된다.
  <script>
    alert("내장형");
  </script>
</head>
<body>
  <button onclick="alert('인라인형');">클릭</button>
</body>
</html>
```

```
//alert.js 파일 내용은 다음과 같다.
//html과 같은 위치에 alert.js파일을 만들면된다.
alert('외장형1')
alert('외장형2')
```

자바스크립트는 위에서 아래로 실행되어 외장형1,외장형2, 내장형 순으로 출력이된다.
인라인형은 버튼을 누를 때 마다 실행 된다.

> 03. 상수 변수 그리고 자료형

자료형은 현실세계의 여러 데이터를 자바스크립트에서 쉽게 사용할 수 있도록 미리 만들어 놓은 데이터 형태이다. JavaScript에서는 여러 종류의 데이터를 다루기 위해 다양한 자료형을 제공합니다. 주요 자료형 (Data Types)으로는 boolean, 숫자, 문자, 배열, 객체 등이 있습니다.

javascript에서 데이터를 표현할때 사용하는 방법은 변수와 상수가 있다. 상수는 값을 변경할 수 없는 데이터를 의미한다. 변수는 원하는 데이터를 저장해 두었다가 프로그램 중간에 값을 변경하거나 읽어올 일 이 생길때 사용한다.

다음 상수 기술 방법이다.

숫자(Number) 상수: 3, 4 ,10, 123.2 정수와 실수로 구성되어 있다.

문자열(String) 상수: ' ' 또는 ""로 묶어서 문자열을 표현한다. '123', "안녕"

불린(Boolean) 상수: true: 참(참값을 나타내는 상수) false: 거짓(거짓값을 나타내는 상수)

다음 변수 기술 방법이다.

JavaScript에서 변수는 자료형에 관계없이 var 키워드로 선언합니다. JavaScript는 동적으로 자료형을 결정하므로 변수에 값이 할당되면 해당 값의 자료형을 결정합니다.

```
var a=10;                var b= "hello";    //var a; a=10; 변수선언 및 값 할당
document.write(a);       document.write(b);//변수를 통한 값 출력
document.write(10);      document.write("world");//상수를 통한 값 출력
a= "hello";              b=10;              //변수의 자료형과 값을 변경
document.write(a);       document.write(b);//변경된 변수의 값 출력
```

예를 들어, var a = 10;은 변수 a에 정수 데이터를 저장하여 a변수의 자료형은 number가 되며, var b = "hello";는 변수 b에 문자열 데이터를 저장하여 자료형이 string이 됩니다.

변수에 원하는 데이터를 할당하면 변수를 할당된 데이터와 같은 취급을 한다.

a=10; 이라 하면 a 에 10이 들어가 document.write(a)은 document.write(10)과 같은 10이 되고 a=20;를 넣고 a값을 찍으면 20이 출력되고 a의 자료형은 number가 된다.

a= "hello"; 이라 하면 a 에 "hello"가 들어가 document.write(a)의 값은 "hello"가 되고 a= "world";를 넣고 a값을 찍으면 "world"가 출력되고 a의 자료형은 string이 된다.

상수는 값을 변경할 수 없지만 변수는 할당 연산자를 이용해서 값을 변경할 수 있다.

10=11은 불가능하고 a=11은 가능하다.

자바스크립트에 다양한 자료형이 존재하는데 자바스크립트 문법에서 제공하는 자료형 선언에 사용하는 자료형을 담는 키워드는 var 하나만 존재 한다. 그래서 모든 자료형을 var에 넣어서 처리 한다. 하지만, 개발자는 해당 변수가 어떤 자료형인지 알고 있어야 한다.

자바스크립트에 있는 자료형을 살펴 보자.

typeof는 다음에 오는 데이터의 자료형이 어떤 자료형인지 문자열이 생성된다.

숫자형 (Number): 정수와 실수를 다루는 자료형입니다.

```
document.write(10, typeof(10), "<br>"); // 출력: 10 number
document.write(10.1, typeof(10.1), "<br>"); // 출력: 10.1 number
var myNum = 10;
document.write(myNum, typeof(myNum), "<br>"); // 출력: 10 number
myNum = 10.1;
document.write(myNum, typeof(myNum), "<br>"); // 출력: 10.1 number
```

불리언 (Boolean):

true와 false 두 가지 값을 가지는 자료형입니다.

```
var myBoolean = true;
document.write(myBoolean, typeof(myBoolean), "<br>"); // 출력: true boolean
myBoolean = false;
document.write(myBoolean, typeof(myBoolean), "<br>"); // 출력: false boolean
```

문자열 (String):

작은 따옴표 (') 또는 큰 따옴표 (")로 묶어서 문자열을 표현합니다.

```
var myString = "hello";
document.write(myString, typeof(myString), "<br>"); // 출력: hello string
```

배열 (Array):

여러 값을 하나의 변수에 저장하는 자료형입니다. 대괄호 []를 사용하여 정의하며, 각 요소는 쉼표로 구분됩니다.

```
var myArray = ["A", "B", "C"];
document.write(myArray[0], "<br>"); // 출력: A
myArray[0] = "D";
document.write(myArray[0], "<br>"); // 출력: D
document.write(myArray, typeof(myArray), "<br>"); // 출력: [D, B, C] object
```

객체 (Object): 자바스크립트 객체를 json이라고 한다.

키-값 쌍으로 구성된 복합 자료형입니다. 중괄호 { }를 사용하여 정의합니다.

```
var myPerson = {
  firstName: "John",
  age: 40,
```



```

    eyeColor: "blue"
};
document.write(myPerson.firstName, myPerson.age, myPerson.eyeColor, "<br>");
// 출력: John 40 blue
myPerson.age = 50;
document.write(myPerson.firstName, myPerson.age, myPerson.eyeColor, "<br>");
document.write(myPerson, typeof(myPerson), "<br>"); //John 50 blue object
객체의 내용을 명확히 알고 싶다면 console.log로 찍으면 자세히 확인할 수 있다.

```

함수 (Function):

함수는 반복되는 코드 블록을 미리 정의해두고, 필요할 때 해당 블록을 호출하여 실행하는 방법입니다. 코드의 선언부는 반복되는 작업을 수행하는 코드 블록을 의미하며, 호출부는 선언된 코드 블록을 실행시키는 부분입니다. 이러한 호출은 필요할 때마다 반복됩니다. 함수를 사용하면 코드를 논리적으로 나누고, 재사용성을 높일 수 있어서 코드를 더 효율적으로 관리할 수 있습니다

```

var myFunction = function () { //선언부
    alert('hello');    alert('world');
};
document.write(myFunction, typeof(myFunction)); // 출력: function
myFunction(); // 'hello' 알림이 표시됨 //호출부
myFunction();
myFunction();

```

기타 사항:

undefined: 자료형이 결정되지 않고 값이 할당되지 않은 상태를 나타냅니다.

```

var a;
document.write(a, typeof(a), "<br>"); // 출력: undefined

```

null: 값은 할당되지 않았지만 자료형이 object인 특수한 상태를 나타냅니다.

```

document.write(b, typeof(b)); // 출력: null

```

다음 코드와 코드 설명을 읽어보고 확인해 보자.

1.

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>기초3</title>
    <script>
      //숫자형(Number)
      document.write(10, typeof(10), "<br>"); //출력 10 number
      document.write(10.1, typeof(10.1), "<br>"); //출력 10.1 number
      var myNum = 10;
      document.write(myNum, typeof(myNum), "<br>"); //출력 10 number
      myNum = 10.1;
      document.write(myNum, typeof(myNum), "<br>"); //출력 10.1 number
    </script>
  </head>
</html>

```

```

//불리언(Boolean)
var myBoolean = true;
document.write(myBoolean,typeof(myBoolean),"<br>");//true boolean
myBoolean = false;
document.write(myBoolean,typeof(myBoolean),"<br>");//false boolean

//배열(Array)
var myArray = ["A", "B", "C"];
document.write(myArray[0],"<br>"); //출력 : A
myArray[0] = "D";
document.write(myArray[0], "<br>"); //출력 : D
document.write(myArray, typeof(myArray),"<br>"); //D, B, Cobject

//객체(Object)
var myPerson = {
    fistName: "John",
    age:40,
    eyeColor:"blue"
};
document.write(myPerson.fistName, myPerson.age,
myPerson.eyeColor,"<br>"); //출력 : John40bLue
myPerson.age = 50;
document.write(myPerson.fistName, myPerson.age,
myPerson.eyeColor,"<br>"); //출력 : John50bLue
document.write(myPerson,typeof(myPerson),"<br>");//object object

//함수(Function)
var myFunction = function(){
    alert('hello');
};
document.write(myFunction, typeof(myFunction),"<br>"); //출력 :
function(){alert('hello');}function
myFunction(); // 'hello' 알림이 표시 됨

//기타 사항
//undefined: 자료형이 결정되지 않고 값이 할당되지 않은 상태
var a;
document.write(a, typeof(a),"<br>"); //출력 : undefinedundefined

//null : 값은 할당되지 않았지만 자료형이 object인 특수한 상태
var b=null;
document.write(b, typeof(b)); //출력: null Object
</script>
</head>
<body>

```

```
</body>
</html>
```

2. <!DOCTYPE html>

```
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script>
    var a=1;
    var a=2;//기존선언된 a값은 사라지고 새로 선언한 a값만 남는다.
    document.write(a);//2값만 남음
    document.write(10/6,"<br>");//자바스크립트는 자동으로 실수처리한다. 1.666
    document.write(10/2.);//5 정수로 출력
```

//자바스크립트에서 \n은 줄바꿈 \t는 탭을 \"은 “를 문자열로 출력을 의미하지만 브라우저에서는 적용되지 않고 alert에서는 적용된다.

```
document.write("안\n녕\nhello\" <br>");
// alert("안\n녕\nhello<br>");
// alert("안녕 \" hello<br>");
// alert('안녕 " hello<br>');
// alert("안녕 ' hello<br>");
```

//+연산은 앞에서 2개씩 실행되어 숫자면 연산되어 연산결과가 나오고 둘중에 하나라도 문자열이면 문자열로 변경되어 문자열이 합쳐진다.

```
document.write("11+22+33","<br>");//11+22+33
document.write(11+22+"33","<br>");//3333
document.write("11"+22+33,"<br>");//112233
document.write("11"+"22"+"33","<br>");//112233
document.write("11"+"22"+"33","<br>");//112233
```

//boolean 자료형을 숫자로 형변환하면 true 는1 false 는0 으로 변환된다.

//자바의 경우 컴파일 에러가 난다.

```
document.write(true+1,"<br>");//2 //false 0 true 1
document.write(false+1,"<br>");//1 //false 0 true 1
document.write(true+"1","<br>");//true1
```

//일바 자료형을 boolean형으로 형변환하면 0이면 false 0이 아니면 true 이다.

```
document.write(0,"<br>");//0
document.write(Boolean(0),"<br>");//false Boolean()은 Boolean형변환 의미
// 0은 boolean으로 형변환되면 false이다. 0은 false로 자동 형변환 되었다가
```

```

    //!로 true가 되었다가 !으로 다시 false가 되었다.
    document.write(!!0,"<br>");//false

    document.write(10,"<br>");//10
    document.write(Boolean(10),"<br>");//true
    document.write(!!10,"<br>");//true
    //'hi'                                //true
    //0, NaN, '', null , undefined      //false
</script>
</head>
<body>
</body>
</html>

```

코드 설명

`백틱으로 문자열을 감싸면 일반적인 문자열과 동일한 문자열을 생성할 수 있습니다.

```

const name = "Alice";
const greeting = `안녕하세요, ${name}!`;
console.log(greeting); // "안녕하세요, Alice!"

```

여러 줄에 걸친 문자열을 쉽게 생성할 수 있습니다.

```

const multiLineText = `
    이것은
    멀티라인
    문자열입니다.
`;
console.log(multiLineText);
/*
    이것은
    멀티라인
    문자열입니다.
*/

```

`${}`를 사용하여 변수나 표현식을 문자열 중간에 삽입할 수 있습니다.

```

const num1 = 5;
const num2 = 10;
const result = `더하기: ${num1 + num2}`;
console.log(result); // "더하기: 15"

```

나중에 사용할 `e1`과 출동이 날 수 있다.

> 04. 다양한 연산자

1. 산술 연산자 (Arithmetic Operators): 수학적 계산을 실행하는 연산

+ : 덧셈 - : 뺄셈 * : 곱셈 / : 나눗셈 % : 나머지 연산 ** : 거듭제곱

```
var addition = 5 + 3; // 5 + 3 = 8
```

```
var subtraction = 10 - 4; // 10 - 4 = 6
```

```
var multiplication = 6 * 2; // 6 * 2 = 12
```

```
var division = 20 / 4; // 20 / 4 = 5
```

```
var modulus = 17 % 5; // 17 % 5 = 2
```

```
var exponentiation = 2 ** 3; // 2^3 = 8
```

2. 할당 연산자 (Assignment Operators): 변수에 값을 할당하는 연산

= : 값 할당 += : 덧셈 후 할당 -= : 뺄셈 후 할당

*= : 곱셈 후 할당 /= : 나눗셈 후 할당 %= : 나머지 연산 후 할당

```
var x = 10; //x에 10을 넣는다.
```

```
x += 5; // x = x + 5 = 15   계산된 결과를 x에 넣는다.
```

```
var y = 20; //y에 20을 넣는다.
```

```
y -= 8; // y = y - 8 = 12
```

3. 비교 연산자 (Comparison Operators): 두 개의 값을 비교하여 조건의 참 또는 거짓을 판단하는 연산자 연산결과 boolean자료형의 true false가 생성된다.

== : 값이 같은지 비교 (동등 연산자)

!= : 값이 다른지 비교 (부등 연산자)

=== : 값과 타입이 모두 같은지 비교 (일치 연산자)

!== : 값 또는 타입이 다른지 비교 (불일치 연산자)

> : 크다 < : 작다 >= : 크거나 같다 <= : 작거나 같다

'1'==1 '1'은 형변환 한 값이 1이므로 true이고 '1'===1은 자료형이 다르므로 false이다.

```

var x=15; var y=10;

var isEqual = x == y; // false

var isNotEqual = x != y; // true

var isStrictEqual = x === 15; // true

var isNotStrictEqual = y !== '12'; // true

var isGreater = x > y; // true

var isLess = x < y; // false

var isGreaterOrEqual = x >= y; // true

var isLessOrEqual = y <= 12; // true

```

4. 논리 연산자 (Logical Operators):

두 개의 boolean 자료형 값을 연산하여 하나의 결과를 얻는 연산자에는 여러 종류가 있습니다. 일반적으로, 두 조건이 모두 true일 때 true를 반환하는 && 연산자, 둘 중 하나만 true여도 true를 반환하는 || 연산자, 그리고 true면 false, false면 true로 반전시키는 ! 연산자가 있습니다. 이 연산자들을 사용하여 boolean 값을 조작할 수 있습니다.

진리표를 웹에서 검색해서 좀더 자세한 연산 결과를 확인해 보자.

&& : 논리적 AND (그리고) || : 논리적 OR (또는) ! : 논리적 NOT (부정)

```

var condition1 = true;

var condition2 = false;

var logicalAnd = condition1 && condition2; // false

var logicalOr = condition1 || condition2; // true

var logicalNot = !condition1; // false

!false===true 의 결과는 true가 된다.

```

5. 삼항 연산자 (Ternary Operator):

condition ? “condition이 true일때실행”: “condition이 false일때실행”; condition이
 상태에 따라 두 가지 표현식 중 하나가 선택이 된다.

```

var age = 18;

```

```
var isAdult = age >= 18 ? '성인' : '미성년자'; // '성인'
```

6. 증감 연산자

증감 연산자는 변수의 값을 1만큼 증가(++)시키거나 1만큼 감소(--)시키는 데 사용

증감 연산자의 위치에 따라 연산 결과가 달라질 수 있다.

전치 연산 (++count, --count): 변수의 값을 증가시키거나 감소시킨 후, 그 값을 반환합니다.
`var a = 5;`

`var b = ++a;` // a의 값을 1 증가시킨 후, b에 6을 할당

후치 연산 (count++, count--): 변수의 값을 반환한 후, 그 값을 증가시키거나 감소시킵니다.

`var x = 10;`

`var y = x--;` // x의 값을 y에 할당한 후, x의 값을 1 감소시킴 실행후 x=11,y=10

다음 예제를 실행해 보자.

//관계연산자, 비교연산자

```
document.write(50>60,"<br>");
document.write(11=='11',"<br>");
document.write(11==='11',"<br>");
document.write(11!='11',"<br>");
document.write(11!== '11',"<br>");
document.write(7>6>5,"<br>");//false
document.write(''==false,"<br>");//true
document.write(''==0,"<br>");//true
document.write(''===false,"<br>");//false
document.write(''===0,"<br>");//false
str="";
document.write(str!=0,"<br>");//false
document.write(str!==0,"<br>");//true
str="0";
document.write(str!=0,"<br>");//false
document.write(str!==0,"<br>");//true
//산술연산자 +,-,/,*,%
var a=10,b=20;
document.write(a+b,"<br>");//true
document.write(b*3,"<br>");//true
document.write(10%3,"<br>");//true
//관계연산자 || 둘중 하나가 true이면 true && 둘다 true이면 true
//진리표를 웹에 검색해서 확인해보자.
document.write(1>2&& 3>4,"<br>");//true
```

```
document.write(1==2 || 1!=2, "<br>");//true
//조건연산자
//a?b:c; a조건이 참이면 b가 남고 거짓이면 c가 남는다.
var a=(10>20)?10:20;
document.write(a);
```

```
증감연산자.    ++ --
var a=0;
a++;
document.write(a);
a--;
document.write(a);
++a;
document.write(a);
--a;
document.write(a);
```

```
document.write(a++);//0
document.write(a);//1
document.write(++a);//2
document.write(a);//2
```

```
//대입연산
a=10;
a+=10;//a=a+10;
a-=10;//a=a-10;
```

```
//연산자 우선순위
//생각과 다른 결과가 나오면 웹에서 연산자 우선순위를 확인해보자.
```

```
document.write(5>6==7>8, "==연산자보다 비교 연산자가 우선순위가 높다<br>");//2
document.write(5==6&&7>8, "논리연산자가 비교연산자보다 우선순위가 낮다.<br>");//2
```

> 05. 내장함수

javascript에서 앞에 내장이 붙으면 전문 프로그래머가 미리 만들어 놓은 코드들을 의미이다.

함수는 필요한 코드를 미리 만들어 놓고 사용하고 싶을 때 가져다가 사용하는 코드 블록을 의미한다. 내장함수는 전문 프로그래머가 미리 구현해 놓은 함수를 사용하는 것이다. 다음과 같은 내장함수를 배울 예정이다. prompt(),alert(),confirm()

//prompt를 이용해서 입력창을 통해 사용자 입력을 입력받을 수 있다.

// prompt코드 부분에서 사용자 입력을 할 수 있도록 팝업이 뜬다. 원하는 성적을 입력하고 프로그램을 진행하면 사용자 입력을 프로그램 안에서 사용할 수 있다.

//alert를 이용해서 문자열을 팝업으로 출력할 수 있다.

//confirm true, false를 선택해서 둘중에 하나의 결과를 만들어 준다.

//document.write 브라우저에 출력하는 함수이다.

//log.console콘솔창에 출력하는 함수이다

//Number(),parseFloat(),parseInt(),String(),toFixed()

//Number는 문자열을 숫자 데이터로 변경한다.

//parseFloat는 문자열을 실수로 변환한다.

//parseInt는 문자열을 숫자로 변환한다.

//String 다른 자료형을 문자열로 변환한다.

//toFixed(2) 실수를 소수점 2자리로 반올림 한다.

var result=prompt("성적은?", "여기입력");

*document.write(result, "
");*

*document.write(result+10, "
");//사용자입력은 문자열*

*document.write(typeof result, "
");*

result=Number(result);

*document.write(result+10, "
");*

*document.write(typeof result, "
");*

var result='14.5';

*document.write(Number(result), "
");*

*document.write(parseFloat(result), "
");*

*document.write(parseInt(result), "
");*

*document.write(String(result), "
");*

var result='14';

*document.write(Number(result), "
");*

*document.write(parseFloat(result), "
");*

*document.write(parseInt(result), "
");*

*document.write(String(result), "
");*

var result='14.5원';

*document.write(Number(result), "
");*

*document.write(parseFloat(result), "
");*

*document.write(parseInt(result), "
");*

*document.write(String(result), "
");*

var word=window.confirm("true or false"?true:false;

*document.write(word, "
");*

//toFixed

var number = 3.14159265;

//소수점 두 자리로 반올림한 문자열 "3.14"

```
var roundedNumber=number.toFixed(2);  
document.write(roundedNumber,"<br>");
```

다음 연습문제를 구현해 보자.

- 1.사용자로부터 2개의 정수를 받아서 첫 번째 정수를 두 번째 정수로 나누었을 때의 몫과 나머지를 계산하는 프로그램을 작성하라.
- 2.사용자로부터 하나의 정수를 받아서 정수의 세제곱 값을 계산하여 출력하는 프로그램을 작성하라.
- 3.사용자로부터 3개의 정수를 받아서 변수 x, y, z에 저장하고 다음 수식 ($x * y - z$)의 결과를 출력하는 프로그램을 작성하라.
- 4.세 자리로 이루어진 숫자를 입력받은 후 각각의 자리수를 분리하고 이 자리수를 출력하는 프로그램을 작성하라.
- 5.다음 수식의 값을 계산하여 화면에 출력하라. x의 값은 사용자로부터 입력받는다.
 $f(x) = (x^3 - 20) / (x - 7)$
- 6.2차원 평면에서 두 점 사이의 거리를 계산하는 프로그램을 작성한다.

답안

1.

```
var input1 = parseInt(prompt("첫 번째 정수를 입력하세요:"));  
var input2 = parseInt(prompt("두 번째 정수를 입력하세요:"));  
var quotient = Math.floor(input1 / input2);  
var remainder = input1 % input2;  
alert("몫은 " + quotient + " 나머지는 " + remainder);
```

2.

```
var input = parseInt(prompt("정수를 입력하세요:"));  
var result = input * input * input;  
alert(input + "의 세제곱 값은 " + result + "입니다.");
```

3.

```
var x = parseInt(prompt("첫 번째 정수를 입력하세요:"));  
var y = parseInt(prompt("두 번째 정수를 입력하세요:"));  
var z = parseInt(prompt("세 번째 정수를 입력하세요:"));  
var result = x * y - z;  
alert("결과는 " + result + "입니다.");
```

4.

```
var number = parseInt(prompt("세 자리 숫자를 입력하세요:"));  
var hundreds = Math.floor(number / 100);  
var tens = Math.floor((number % 100) / 10);
```

```
var ones = number % 10;
alert(hundreds + "백 " + tens + "십 " + ones);
```

5.

```
var x = parseFloat(prompt("x의 값을 입력하세요:"));
var result = (Math.pow(x, 3) - 20) / (x - 7);
alert("f(x)의 값은 " + result.toFixed(4));
```

6.

```
var x1 = parseFloat(prompt("첫 번째 점 x1 입력:"));
var y1 = parseFloat(prompt("첫 번째 점 y1 입력:"));
var x2 = parseFloat(prompt("두 번째 점 x2 입력:"));
var y2 = parseFloat(prompt("두 번째 점 y2 입력:"));
var distance = Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));
alert("거리는 " + distance.toFixed(5));
```

> 06. 조건문

제어문에는 조건문과 반복문이 있다.

조건문: 특정 조건이 참인지 거짓인지에 따라 특정 블록을 실행 시키거나 실행 하지 못하게 할 수 있다.

조건문의 주요 유형으로는 if, if-else, else if 등이 있습니다. 이를 통해 프로그램은 특정 조건에 따라 다른 동작을 수행하도록 분기할 수 있습니다.

반복문: 코드 블록을 반복해서 실행하는 데 사용됩니다. 주로 배열의 각 요소를 순회하거나, 특정 횟수만큼 반복 작업을 수행하는 데 활용됩니다. 주요한 반복문으로는 for, while, do-while 등이 있습니다.

제어문을 적절히 활용하면 프로그램의 실행 흐름을 조작하여 원하는 결과를 얻을 수 있습니다.

JavaScript에서 조건문은 프로그램이 특정 조건에 따라 다른 동작을 수행하도록 하는데 사용됩니다.

주요한 조건문 형태로는 if문, if-else문, else if문이 있습니다.

if 문: 특정 조건이 참일 경우에 코드 블록을 실행합니다.

```
if (조건) {  
    // 조건이 참일 때 실행되는 코드  
}
```

```
var temperature = 25;  
if (temperature > 30) {  
    console.log("더워요!");  
}
```

위의 예제에서, temperature 변수의 값이 30보다 크면 "더워요!"라는 메시지가 출력됩니다.

if-else 문: 특정 조건이 참이면 첫 번째 코드 블록을 실행하고, 그렇지 않으면 두 번째 코드 블록을 실행합니다.

```
if (조건) {  
    // 조건이 참일 때 실행되는 코드  
} else {  
    // 조건이 거짓일 때 실행되는 코드  
}
```

```
var age = 17;  
if (age >= 18) {  
    console.log("성인입니다.");  
} else {  
    console.log("미성년자입니다.");  
}
```

위의 예제에서, age 변수의 값이 18 이상이면 "성인입니다."라는 메시지가 출력되고, 그렇지 않으면 "미성년자입니다."라는 메시지가 출력됩니다.

else if 문: 여러 개의 조건을 동시에 체크하여 위에서 아래로 처음 조건에 맞는 하나의 코드 블록을 실행한다. 일치하는 것이 없으면 else에 기술된 코드 블록이 실행 된다.

```
if (조건1) {  
    // 조건1이 참일 때 실행되는 코드
```

```

} else if (조건2) {
    // 조건2가 참일 때 실행되는 코드
} else {
    // 위의 모든 조건이 거짓일 때 실행되는 코드
}

```

```

var score = 75;
if (score >= 90) {
    console.log("A 학점");
} else if (score >= 80) {
    console.log("B 학점");
} else if (score >= 70) {
    console.log("C 학점");
} else {
    console.log("D 학점");
}

```

위의 예제에서, score 변수의 값에 따라 다른 학점이 출력 됩니다.

switch문 : else if문중 비교 대상이 1:1 매칭이면 switch문을 만들 수 있다.

```

switch (변수) {
    case 값1:
        // 표현식이 값1과 일치할 때 실행되는 코드
        break;
    case 값2:
        // 표현식이 값2와 일치할 때 실행되는 코드
        break;

    // 추가적인 case문들...
    default:
        // 모든 case에 해당하지 않을 때 실행되는 코드
}

```

switch 뒤 () 중괄호 안에 비교할 변수를 넣는다.

각 case는 비교할 변수의 값과 일치할 때 실행될 코드 블록입니다.

break 문은 각 case 블록의 끝에서 사용되어 해당 case에서 코드 실행을 중단하고 switch문을 빠져나갑니다.

case문 안에 break문이 없으면 다음 case문이 실행된다.

default는 어떤 case에도 일치하지 않을 때 실행될 코드 블록입니다. default는 선택사항이며 생략할 수 있습니다.

```

var day = "월요일";
switch (day) {
    case "월요일":
        console.log("열심히 시작하는 월요일!");
        break;
    case "수요일":
        console.log("이제 절반을 향해 가는 수요일!");
        break;
}

```

```

case "금요일":
    console.log("주말이 곧 다가오는 금요일!");
    break;
default:
    console.log("일주일의 중간일 것 같습니다.");
}

```

조건문을 사용하여 프로그램이 상황에 따라 올바른 동작을 수행하도록 제어할 수 있습니다.

다음 문제를 풀어 보자.

1. 사용자에게 숫자를 입력받아 입력받은수가 100보다 큰지 작은지 출력
2. 100보다 작으면 홀수 인지 짝수 인지 100보다 크면 3의 배수인지 결과를 출력하시오. 1.

```

var a=5;
    if(a>100){
        alert('크다');
    }else{
        alert('100보다 작다');
    }
2.    if(a>100){
        if(a%3==0){
            alert('3의 배수');
        }else{
            alert('3의 배수가 아님');
        }
    }else{
        if(a%2==0){
            alert('짝수');
        }else{
            alert('홀수');
        }
    }
}

```

다음 문제를 풀어 보자.

1. 현재시간이 새벽, 아침, 점심, 저녁 중 언제인지 확인 하시오. 시간대는 24시간을 4로 나눠 표시 다음을 if else if 문에서 switch 문으로 변경해 보자.
2. 사용자에게 숫자를 입력 받아 양수, 0, 음수를 구분하는 프로그램
3. 사용자에게 숫자를 입력 받아 홀수와 짝수를 구분하는 프로그램
4. 사용자에게 국어, 영어, 수학 점수를 입력 받아 평균을 구한 뒤 수우미양가를 구분하는 프로그램

```

var kor=Number(prompt("kor"));
var eng=Number(prompt("eng"));
var math=Number(prompt("math"));
var avg=(kor+eng+math)/3;
document.write(avg);
//.toFixed(2);
document.write('<br>',avg.toFixed(2));
if(avg>90){

```

```

        alert("수");
    }else if(avg>80){
        alert("우");
    }else if(avg>70){
        alert("미");
    }else if(avg>60){
        alert("양");
    }else{
        alert("가");
    }
}

```

//사용자 입력을 받아 홀수 이면 홀수 짝수이면 짝수를 화면에 출력하시오.

```

var input =prompt("숫자입력");
input =Number(input);
if(input%2==0){//+이외의 연산은 자동형변환된다.
    document.write("짝수<br>");
}
if(input%2==1){
    document.write("홀수<br>");
}
if(input%2==0){//+이외의 연산은 자동형변환된다.
    document.write("짝수<br>");
}else{
    document.write("홀수<br>");
}
}

```

//3개중 가장 큰수를 출력

```

var a=512;          var b=52;          var c=30;
if(a>b){
    if(a>c){
        alert(a);
    }else{
        alert(c);
    }
}else{
    if(b>c){
        alert(b);
    }else{
        alert(c);
    }
}
}

```

> 07. 반복문

자바스크립트에서 반복문은 일련의 코드 블록을 여러 번 실행하는 데 사용되는 구문입니다. 주로 배열의 요소를 반복하거나 조건을 만족할 때까지 코드를 실행하는 데 활용됩니다. 여기서 자바스크립트의 세 가지 주요 반복문에 대해 설명하겠습니다.

for 문:

for 문은 초기화, 조건 검사 및 반복 단계로 구성됩니다. 아래는 기본적인 for 문의 구조입니다:

```
for (초기화; 조건; 반복 단계) {  
    // 실행할 코드  
}
```

예를 들어, 1부터 5까지의 숫자를 출력하는 for 반복문은 다음과 같이 작성할 수 있습니다:

```
for (var i = 1; i <= 5; i++) {  
    console.log(i);  
}
```

while 문:

while 문은 주어진 조건이 참인 동안 코드 블록을 계속 실행합니다. 아래는 while 문의 구조입니다:

```
while (조건) {  
    // 실행할 코드  
}
```

예를 들어, 1부터 5까지의 숫자를 출력하는 while 반복문은 다음과 같이 작성할 수 있습니다:

```
var i = 1;  
while (i <= 5) {  
    console.log(i);  
    i++;  
}
```

do...while 문:

do...while 문은 코드 블록을 먼저 실행하고 조건을 검사합니다. 조건이 참인 경우 코드 블록을 계속 실행합니다. 아래는 do...while 문의 구조입니다:

```
do {  
    // 실행할 코드  
} while (조건);
```

do...while 문은 코드 블록을 최소한 한 번은 실행하므로 조건이 false일 때까지 반복을 보장합니다.

예를 들어, 1부터 5까지의 숫자를 출력하는 do...while 반복문은 다음과 같이 작성할 수 있습니다:

```
var i = 1;  
do {  
    console.log(i);  
    i++;  
} while (i <= 5);
```

이러한 반복문은 다양한 상황에서 사용되며, 배열 순회, 조건을 충족하는 동안 작업 수행 등 다양한 반복 작업을 수행하는 데 유용합니다.

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        //4~8까지 모든 수를 더하는 반복문  
        var sum=0;
```



```

        for(var i=4;i<=8;i++){
            sun=sum+i;
        }
        document.write(sum);
        var i=4;
        var sum=0;
        while(i<=8){
            sun=sum+i;
            i++;
        }
        document.write(sum);
        var i=4;
        var sum=0;
        do{
            sum=sum+i;
            i++;
        }while(i<=8);
        document.write(sum);
// 1~100까지 정수 중 3의 배수의 합
var sum =0;
for(var i=1;i<=100;i++){

    if(i%3==0){
        sum=sum+i;
    }
}
alert(sum);
//1~100까지 정수 중 사용자가 입력한 수의 배수 합을 구하시요
var input =4;
var sum =0;
for(var i=1;i<=100;i++){
    if(i%input==0){
        sum=sum+i;
    }
}
alert(sum);
//1~100까지의 수중에 3과 4의 배수의 합을 구하시오,
var sum =0;
for(var i=1;i<=100;i++){
    if(i%3==0 || i%4==0){
        sum=sum+i;
    }
}
alert(sum);
//1~100까지의 수중에 3과 4의 공배수의 합을 구하시오
var sum =0;
for(var i=1;i<=100;i++){

```

```

    if(i%3==0&& i%4==0){
        sum=sum+i;
    }
}
alert(sum);

```

자바스크립트에서 break와 continue는 반복문 (예: for, while, do...while) 내에서 제어 흐름을 조작하는데 사용되는 특별한 키워드입니다.

break:

break는 반복문을 조기에 종료하고 반복을 중단할 때 사용됩니다. 주로 조건을 충족하면 반복문을 종료하고 빠져나오는 데 사용됩니다.

예를 들어, 다음은 for 반복문을 사용하여 1부터 5까지의 숫자를 출력하며, if 문을 사용하여 특정 조건에서 반복문을 종료하는 방법을 보여줍니다:

```

for (var i = 1; i <= 5; i++) {
    console.log(i);
    if (i === 3) {
        break; // i가 3일 때 반복문 종료
    }
}

```

위 코드는 1, 2, 3까지만 출력하고 반복문을 종료합니다.

continue:

continue는 현재 반복을 중단하고 다음 반복 단계로 건너뛰는 데 사용됩니다. 주로 특정 조건을 만족하는 경우 현재 반복 단계를 건너뛰고 다음 반복으로 진행하는 데 사용됩니다.

예를 들어, 다음은 for 반복문을 사용하여 1부터 5까지의 숫자 중에서 홀수만 출력하는 방법을 보여줍니다:

```

for (var i = 1; i <= 5; i++) {
    if (i % 2 === 0) {
        continue; // 짝수일 경우 현재 반복 건너뛰
    }
    console.log(i);
}

```

위 코드는 홀수인 숫자 (1, 3, 5) 만 출력하고 짝수는 건너뛸니다.

break와 continue는 반복문 내에서 조건에 따라 코드 흐름을 제어할 때 유용하게 사용됩니다. 이를 통해 특정 조건에 따라 반복을 종료하거나 특정 단계를 건너뛸 수 있습니다.

> 08. 함수

함수 : 반복되는 코드를 미리 기술해 놓고 필요할때 호출해서 사용하는 코드블록

함수선언부, 호출부 : 함수를 사용하려면 함수선언부를 구현하고 사용하고 싶을 때마다 호출부를 이용해서 호출하여 사용한다.

```

선언부
function 함수명(){
    함수내용구현
}
호출부
함수명();

```

다음 매개변수와 리턴값이 없는 함수들을 실행 해보자.

1.

```

function greet() {
    console.log("Hello!");
}
greet(); // 출력: Hello!
greet(); // 출력: Hello!
greet(); // 출력: Hello!

```

2.

```

function printCurrentTime() {
    var currentTime = new Date();
    console.log("Current time is: " + currentTime.toLocaleTimeString());
}
printCurrentTime(); // 출력: 현재 시간

```

3.

```

function getRandomNumber() {
    return Math.random();
}
var randomValue = getRandomNumber();
console.log("Random number: " + randomValue);

```

함수 선언부를 자세히 살펴보자.

```

function functionName(parameter1, parameter2, ...) {
    // 함수 내부에서 실행될 코드
    // return 반환할 값 (optional)
}

```

function: 함수를 정의할 때 사용하는 키워드입니다.

functionName: 함수 이름은 여러 함수를 구분하기 위해서 사용한다. 함수 이름은 유효한 식별자(영문, 숫자로 시작)여야 합니다. 소문자로 시작해서 새로운 의미 단어가 나올때 마다 대문자로 기술한다.

parameter1, parameter2, ...: 매개변수(parameter)에 해당하며, 함수 내에서 사용할 변수에 해당하고 함수 호출 시 소괄호 안에 매개 변수 값을 넣어 호출 한다.

{ } 내부: 종괄호로 감싸진 부분은 함수의 본문이며, 실제로 실행되는 코드입니다.

함수 내에서 원하는 작업을 수행

return: 함수 실행후 호출한 부분으로 전달할 값이 있으면 return 키워드를 이용해서 함수 종료후 원하는 값을 반환할 수 있다. return 다음에 전달할 데이터를 기술 하면된다. 전달할 값이 없으면 생략 가능하다. return 10; 함수 종료후 10이 전달된다. return "hello" 함수 종료후 hello 문자열이 전달된다. return; 아무것도 전달되지 않는다. return 자체를 생략할 수 있다.

정의한 함수를 호출하려면 함수 이름을 사용하고 필요한 인수(인자)를 전달합니다. 함수 호출시

매개변수에 값을 데이터를 인자라 한다. 보통 편이상 인자도 매개변수라 부른다.

매개변수와 리턴 값이 있는 함수 사용방법

함수를 호출하여 리턴값 받는 방법

```
const result = functionName(arg1, arg2, ...);
```

result: 함수가 반환하는 결과 값이 저장될 변수

함수가 값을 반환하지 않는 경우 result에는 undefined가 할당

functionName: 호출할 함수의 이름입니다.

arg1, arg2, ...: 함수에 전달할 인수(argument)를 나열합니다. 매개 변수라 부르기도 한다.

함수 정의에서 정의한 매개변수와 순서와 수를 일치시켜 호출 해야 한다.

간단한 덧셈 함수 예:

```
function add(a, b) {    //함수 선언
    return a + b;      //리턴 값
}
```

```
const sum = add(2, 3); // 함수 호출
console.log(sum);      // 5 출력
console.log(add(5,5)); // 10 출력 리턴값을 바로 사용
```

위 예제에서 add 함수는 두 개의 인수를 받아서 합을 반환합니다. 함수를 호출하고 그 결과를 sum 변수에 저장하고, console.log로 결과를 출력합니다.

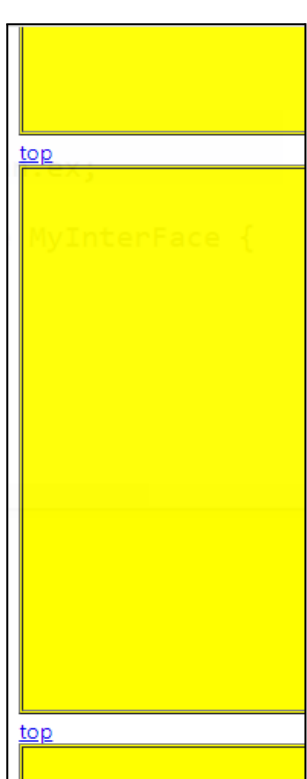
```
// 함수를 변수에 할당하여 사용할 수 있다.
var sum = add;
// 변수를 통해 함수 실행
var result = sum(3, 5); // 함수를 호출하면 result에 8이 저장됨
```

다음 예제들을 확인해 보자.

```
function repeatString(str, count) {
    for (var i = 0; i < count; i++) {
        console.log(str);
    }
}
repeatString("Hello", 3);
// 출력:// Hello// Hello// Hello
2.
function calculateSum(numbers) {
    var sum = 0;
    for (var i = 0; i < numbers.length; i++) {
        sum += numbers[i];
    }
    console.log("Sum: " + sum);
}
calculateSum([1, 2, 3, 4, 5]); // 출력: Sum: 15
3.
```

```
function repeatString(str, count) {
    var repeatedString = "";
    for (var i = 0; i < count; i++) {
        repeatedString += str;
    }
    return repeatedString;
}
var repeatedText = repeatString("Hello", 3);
console.log("Repeated Text: " + repeatedText);
// 출력: Repeated Text: HelloHelloHello
```

다음 예제를 확인해 보자.



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<title>Document</title>
<script>
//myTop,myTable
function myTable(){
document.write("<table bgcolor='yellow' width ='100%'
height=400px border=1><tr><td></td></tr></table>");
}
function myTop(){
document.write("<a href='#'>top</a>");
}
myTable(); myTop();
myTable(); myTop();
myTable(); myTop();
```

/* 다음 문제 들을 풀어보자.

1. “안녕” 이라는 문자열을 출력하는 hi 함수를 만들어 화면에 안녕을 3개 출력해보자.
2. 매개변수로 받은 문자열을 출력하는 myPrint 함수를 만들어 보자.
3. 문자열과 숫자를 입력받아서 해당 문자열을 숫자만큼 반복해서 화면에 찍어보자.
4. 500원짜리랑 100원짜리 개수를 입력받아 금액을 리턴해주는 calChange 메소드를 만들어

보자.

*/

```
function hi(){
    document.write("안녕");
}
hi(); hi(); hi();
function myPrint(str){
    document.write(str, "<br>");
}
```

```

myPrint('안녕');
myPrint('방가워');
myPrint('또봐');
function msgCount(msg,count){
    for(var i=0;i<count;i++){
        document.write(msg+"<br>");
    }
}
msgCount('hi~',4);

function calChange(change100,change500){
    var rValue=change100*100+change500*500;
    return rValue;
}
document.write(calChange(5,2));
</script>
</head>
<body>
</body>
</html>

```

> 09. 문자열로 구구단 구현

```

for (var i = 0; i < 3; i++) {
    console.log(i, 0);
    console.log(i, 1);
    console.log(i, 2);
}

```

0	0
0	1
0	2
1	0
1	1
1	2

상위 코드를 실행하면 콘솔에 왼쪽과 같은 결과를 얻을 수 있다.
 잘 생각해 보면 다음 코드를 반복문으로 변경 할 수 있다.

```

console.log(i, 0); console.log(i, 1); console.log(i, 2);
for (var j = 0; j < 3; j++) {

```

```

    console.log(i, j);
}
따라서, 상위 코드는 다음 이중for문과 같은 결과를 얻을 수 있다.
for (var i = 0; i < 3; i++) {
    for (var j = 0; j < 3; j++) {
        console.log(i, j);
    }
}

```

다음과 같은 테이블을 이중 for문을 이용해서 만들어 보자. 어려우면 이중 for문 *찍기를 검색해서 찍어본다음 해보자.

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

1단	2단	3단	4단	5단	6단	7단	8단	9단
1*1=1	2*1=2	3*1=3	4*1=4	5*1=5	6*1=6	7*1=7	8*1=8	9*1=9
1*2=2	2*2=4	3*2=6	4*2=8	5*2=10	6*2=12	7*2=14	8*2=16	9*2=18
1*3=3	2*3=6	3*3=9	4*3=12	5*3=15	6*3=18	7*3=21	8*3=24	9*3=27
1*4=4	2*4=8	3*4=12	4*4=16	5*4=20	6*4=24	7*4=28	8*4=32	9*4=36
1*5=5	2*5=10	3*5=15	4*5=20	5*5=25	6*5=30	7*5=35	8*5=40	9*5=45
1*6=6	2*6=12	3*6=18	4*6=24	5*6=30	6*6=36	7*6=42	8*6=48	9*6=54
1*7=7	2*7=14	3*7=21	4*7=28	5*7=35	6*7=42	7*7=49	8*7=56	9*7=63
1*8=8	2*8=16	3*8=24	4*8=32	5*8=40	6*8=48	7*8=56	8*8=64	9*8=72
1*9=9	2*9=18	3*9=27	4*9=36	5*9=45	6*9=54	7*9=63	8*9=72	9*9=81

역따옴표는 javascript에서 문자열을 만드는 방법중 하나다. e1과 사용할때 문제가 생길 수 있어서 권장하지 않는다.

`${}` 문법은 템플릿 리터럴 (Template Literal)이라고 불리는 특별한 문자열 표기법입니다. `${}`안에 기술된 변수, 상수를 가지고 연산된 결과를 가지고 문자열 자료형을 만들때 사용한다.

```

var i = 10;
console.log(`${1+2}+${i}=${i+3}`); //3+10=13과 같은 문자열이 생긴다.

```

`${1+2}`는 `1 + 2`의 결과인 `3`으로 대체되고, `${i}`는 변수 `i`의 값인 `10`으로 대체됩니다. `${i+3}` 역시 변수 `i`와 `3`을 더한 결과로 대체됩니다. 그 결과 문자열은 다음과 같이 출력됩니다. 다음은 답안이다 확인해 보자.

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <script>
        var str = "";
        str += "<table border=1>";
        for(var j=0; j<5; j++){
            str += "<tr>";
            for(var i=0; i<5; i++){
                str += "<td>"; str += count++; str += "</td>";
            }
        }
    </script>

```

```

    }
    str += "</tr>";
}
str += "</table>";

str += "<table border=1>";
str += "<tr>";
for(var i=1; i<=9; i++){
    str += "<td>";    str += i+"단";    str += "</td>";
}
str += "</tr>";
str += "<tr>";
for(var i=1; i<=9; i++){
    str += "<td>";
    for(var j=1; j<=9; j++){
        str += i+"*" + j + "=" + i*j + "<br>";
    }
    str += "</td>";
}
str += "</tr>";
str += "</table>";

var i = 10;
str += ` ${1+2} + ${i} = ${i+3} `;

document.write(str);
</script>
</body>
</html>

```

> 10. Timer 사용하기

Timer(타이머)는 코드의 실행을 일정 시간 지연하거나 주기적으로 반복하는 데 사용된다.

Timer 함수에는 setTimeout 함수와 setInterval 함수 가 있다. 두 함수는 매개변수로 실행할 함수와 실행될 시간을 받는다.

setTimeout 함수를 사용하면 특정 함수나 코드 블록을 지정된 시간 후에 한 번 실행할 수 있습니다. 지연 시간은 밀리초 단위로 지정됩니다. 1000이 1초이다.

다음 코드는 2초 후에 "안녕하세요!" 메시지를 콘솔에 출력합니다.

```
setTimeout(function() {
```



```
    console.log("안녕하세요!");  
}, 2000); // 2초 후에 실행
```

```
setTimeout(callback, delay);
```

callback: 지연 이후에 실행될 함수(콜백 함수).

delay: 지연할 시간(밀리초).

setInterval 함수를 사용하면 특정 함수나 코드 블록을 일정한 시간 간격으로 주기적으로 실행할 수 있습니다. 간격 또한 밀리초 단위로 지정됩니다. 1000이 1초이다.

다음 코드는 1초마다 현재 시간을 콘솔에 출력합니다.

```
setInterval(function() {  
    var currentTime = new Date();  
    console.log("현재 시간: " + currentTime);  
}, 1000); // 1초마다 실행
```

```
setInterval(callback, interval);
```

callback: 일정한 간격으로 실행될 함수(콜백 함수).

interval: 반복 간격(밀리초).

clearInterval 함수를 사용하여 setTimeout 또는 setInterval 함수로 설정한 타이머를 중지할 수 있습니다. setInterval 함수는 리턴값으로 식별값을 생성한다. 해당 식별값을 변수에 저장해서 clearInterval 함수를 이용해서 생성된 타이머를 삭제 할 수 있다.

두 함수는 각각 setTimeout과 setInterval에서 반환된 타이머 식별자를 사용하여 예약된 작업을 취소합니다.

// setTimeout 취소 예제

```
var timeoutId = setTimeout(function() {  
    console.log("This will not be printed.");  
}, 2000);
```

```
clearTimeout(timeoutId); // setTimeout 취소
```

```
// setInterval 취소 예제
```

```
var intervalId = setInterval(function() {  
    console.log("This will not be printed repeatedly.");  
}, 1000);  
  
clearInterval(intervalId); // setInterval 취소
```

다음 예제는 1초마다 실행되는 setInterval로 만든 타이머를 5초후에 setTimeout안에서 clearInterval함수를 이용해서 종료 한다.

```
var intervalId = setInterval(function() { // setInterval로 1초마다 현재 시간 출력  
    var currentTime = new Date();  
    console.log("현재 시간: " + currentTime);  
}, 1000);  
  
setTimeout(function() { // 5초 후에 타이머 중지  
    clearInterval(intervalId); // 타이머 중지  
    console.log("타이머가 중지되었습니다.");  
}, 5000);
```

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        /*  
        타이머 : 특정 시간에 특정 함수를 실행 시킬때 사용한다.  
        setTimeout 특정시간이후에 1번 실행된다.  
        setInterval 특정시간마다 실행된다.
```

`clearInterval` 특정타이머를 종료시킨다.

```
//Math.random() 0이상 1미만의 실수를 랜덤하게 생성한다.  
//a,b,c변수를 랜덤하게 증가시켜서 먼저 100에 도달하는 순으로  
//등수를 출력해보자.
```

```
타이머이용해서 각각 증가되는 값을 실시간으로 출력  
모두 100이 넘을 때까지 게임이 진행  
모두 도착후 등수 출력*/
```

이 코드는 세 개의 변수를 사용하여 각각 100 이상이 되는 순서대로 등수를 매기는 간단한 경주 게임을 구현한 것입니다. 게임은 타이머를 활용하여 변수를 실시간으로 증가시키고, 100 이상이 되면 해당 변수의 타이머를 중지하며 등수를 기록합니다. 마지막으로 게임이 종료되면 결과를 출력합니다.

변수 초기화 및 등수 기록:

a, b, c 변수는 각각 0으로 초기화되며, endStr 변수는 게임 종료 후 출력될 등수 관련된 문자열을 저장합니다. count 변수는 등수를 기록하기 위한 변수로 1로 초기화되고 각 변수가 등수가 정해지게 될때마다 1씩 증가하게 되어 1,2,3 위가 정해진다.

세 개의 타이머 생성:

setInterval 함수를 사용하여 세 개의 intervalIdA, intervalIdB, intervalIdC 타이머를 생성합니다. 각 타이머는 0.5초(500 밀리초)마다 실행되고 각 타이머마다 a, b, c 의 값을 증가시키다가 중 하나가 100 이상이 되면 해당 타이머를 clearInterval을 사용하여 중지한다.

중지된 타이머의 경우 endStr에 해당 변수의 등수를 추가하고, count를 증가시켜 다음 등수를 기록합니다.

게임 종료 확인 타이머:

intervalIdIsEnd 타이머를 사용하여 게임 종료 여부를 주기적으로 확인합니다.

count가 4가 되면 모든 변수가 100 이상이 되어 게임이 종료되었음을 의미하며, 이때 document.write를 사용하여 게임 종료 메시지와 각 변수의 등수를 출력합니다.

이 코드는 세 개의 변수를 사용하여 각각 100 이상이 되는 순서대로 등수를 매기는 간단한 게임을 구현한 것입니다. 게임은 타이머를 활용하여 변수를 실시간으로 증가시키고, 100 이상이 되면 해당 변수의 타이머를 중지하며 등수를 기록합니다. 마지막으로 게임이 종료되면 결과를 출력합니다.

```
*/  
var a=0,b=0,c=0;  
var endStr='';//게임종료후 출력할 문자열  
var count=1; //등수 기록
```

```

var intervalIdA=setInterval(function(){
    a=a+parseInt(Math.random()*10);
    if(a>=100){
        a=100;
        clearInterval(intervalIdA);
        endStr=endStr+`A는 ${count++}등<br>`;
    }
    document.write("a:"+a+"<br>")
},500)

```

```

var intervalIdB=setInterval(function(){
    b=b+parseInt(Math.random()*10);
    if(b>=100){
        b=100;
        clearInterval(intervalIdB);
        endStr=endStr+`B는 ${count++}등<br>`;
    }
    document.write("b:"+b+"<br>")
},500)

```

```

var intervalIdC=setInterval(function(){
    c=c+parseInt(Math.random()*10);
    if(c>=100){
        c=100;
        clearInterval(intervalIdC);
        endStr=endStr+`c는 ${count++}등<br>`;
    }
    document.write("c:"+c+"<br>")
},500)

```

```

var intervalIdIsEnd=setInterval(function(){
    if(count==4){
        clearInterval(intervalIdIsEnd);
    }
}

```

```

        document.write("<br>게임 종료:<br>" + endStr);
    }
}, 500)
//document.write("시작")
</script>
</head>
<body>

</body>
</html>

```

> 11. 배열과 배열관련 메소드

배열(Array)은 하나의 변수에 여러개의 값을 저장할 때 사용한다. JavaScript에서 배열은 자바의 배열과 list를 합쳐 놓은것과 같은 형태를 가진다.

자바스크립트 배열은 다음과 같은 특징을 가집니다.

순서가 있음: 배열은 각 요소(값)가 인덱스(순서)를 가지며, 이 순서는 0부터 시작합니다.

다양한 형태의 값 저장: 자바 스크립트는 다른언어와 달리 다양한 자료형을 넣을 수 있으나 되도록 동일한 데이터 유형의 값을 저장하여 사용한다.

가변적인 크기: JavaScript 배열은 크기가 동적으로 조절될 수 있으며, 요소를 추가하거나 제거할 수 있습니다.

다양한 메서드와 속성: JavaScript 배열은 다양한 내장 메서드와 속성을 제공하여 배열을

조작하고 처리할 수 있습니다. 예를 들어, push, pop, shift, unshift, length, concat, slice 등의 메서드를 사용할 수 있습니다.

아래는 JavaScript 배열의 기본적인 사용법과 예제입니다:

배열 생성:

```
var fruits = []; // 빈 배열 생성
```

```
var colors = ["빨강", "노랑", "파랑"]; // 초기 값을 가진 배열 생성
```

var a; var b=[];에서 a[2]=1은 a가 배열이 아니므로 오류이고 b[2]=1은 가변 배열이므로 오류가 아니다.

배열 요소 접근:

```
console.log(colors[0]); // "빨강"
```

```
console.log(colors[1]); // "노랑"
```

배열 요소 수정:

```
colors[1] = "주황";
```

```
console.log(colors); // ["빨강", "주황", "파랑"]
```

배열 길이 확인:

```
console.log(colors.length); // 3
```

배열 메서드 사용:

//배열요소 뒤에 추가

```
colors.push("녹색"); // ["빨강", "주황", "파랑", "녹색"]
```

배열에서 요소 제거 뒤에 제거

```
var a=colors.pop(); // "녹색" 제거, ["빨강", "주황", "파랑"]
```

```
console.log(a); //녹색
```

// 배열에 요소 추가 (앞에서)

```
colors.unshift("보라"); // ["보라", "빨강", "주황", "파랑"]
```

// 앞에서 제거

```
var colors = ["빨강", "주황", "파랑"];
```

```
var firstColor = colors.shift(); // firstColor는 "빨강"
console.log(colors); // ["주황", "파랑"] 빨강이 삭제됨
```

배열 메소드 정리

- .concat(): 배열을 합칩니다.
- .every(): 모든 요소가 조건을 만족하는지 확인합니다.
- .filter(): 조건에 맞는 요소만 추출하여 새로운 배열을 생성합니다.
- .indexOf(): 배열에서 특정 요소의 인덱스를 검색합니다.
- .join(): 배열 요소를 문자열로 합칩니다.
- .sort(): 배열 요소를 정렬합니다.
- .reverse(): 배열 요소의 순서를 뒤집습니다.
- .push(): 배열 끝에 요소를 추가합니다.
- .pop(): 배열 끝의 요소를 제거하고 반환합니다.
- .shift(): 배열 앞의 요소를 제거하고 반환합니다.
- .unshift(): 배열 앞에 요소를 추가합니다.
- .slice(): 배열의 일부를 추출하여 새로운 배열을 반환합니다.
- .splice(): 배열에서 요소를 추출하거나 변경합니다.

1차원 배열 순회하는 방법

for 루프 사용:

가장 기본적인 방법으로, 배열의 길이를 사용하여 인덱스를 증가시키면서 배열 요소에 접근합니다.

```
var arr = [1, 2, 3, 4, 5];
for (var i = 0; i < arr.length; i++) {
    console.log(arr[i]);
}
```

for...of 루프 사용:

for...of 루프는 배열의 값을 직접 순회할 때 사용됩니다. 인덱스가 아닌 배열의 요소에 직접 접근합니다.

```
var arr = [1, 2, 3, 4, 5];
for (var item of arr) {
```

```
    console.log(item);  
}
```

forEach() 메소드 사용:

배열 객체의 forEach() 메소드는 배열의 각 요소에 대해 주어진 함수를 호출합니다.

```
var arr = [1, 2, 3, 4, 5];  
arr.forEach(function(item) {  
    console.log(item);  
});
```

map() 메소드 사용:

map() 메소드는 배열의 각 요소에 대한 변환 작업을 수행하고 그 결과를 새로운 배열로 반환합니다.

```
var arr = [1, 2, 3, 4, 5];  
var squaredArr = arr.map(function(item) {  
    return item * item;  
});
```

```
console.log(squaredArr);//[1,4,9,16,25]
```

filter() 메소드 사용:

filter() 메소드는 주어진 조건에 맞는 요소만을 새로운 배열로 필터링합니다.

```
var arr = [1, 2, 3, 4, 5];  
var evenNumbers = arr.filter(function(item) {  
    return item % 2 === 0; //false이면 배열에서 삭제됨  
});
```

```
console.log(evenNumbers);//[2,4]
```

reduce() 메소드 사용:

reduce() 메소드는 배열의 요소를 하나로 축소하여 결과를 반환합니다.

```
var arr = [1, 2, 3, 4, 5];  
var sum = arr.reduce(function(acc, curr) { //배열의 모든 데이터를 합산한 결과 생성  
    return acc + curr;  
}, 0);
```

```
console.log(sum);
```

첫 번째 인수는 누적 작업을 어떻게 수행할지 정의하는 콜백 함수입니다. 이 경우 콜백 함수는 두 개의 매개변수 acc (누산기)와 curr (현재 값)를 받습니다. 이 함수는 현재 값을 누산기에 더합니다.

두 번째 인수는 누산기의 초기 값으로, 여기서는 0으로 설정됩니다.

for...in 루프 사용 (비추천):

for...in 루프를 사용하여 객체의 속성을 순회하는 데에는 사용하지만, 배열 순회에는 적합하지 않습니다. 배열의 인덱스 대신 배열 요소의 키를 반환하기 때문에 주의해야 합니다.

```
var arr = [1, 2, 3, 4, 5];
for (var index in arr) {
    console.log(arr[index]);
}
```

2차원 배열 동적 생성 하는 방법

```
var arr4=[
    [1,2,3],
    [4,5,6],
    [7,8,9],
    [10,11,12]
]
```

//arr4의 배열은 데이터가 총 4개 존재 한다. 배열 데이터 4개

```
document.write(arr4.length, "<br>");//4
```

//arr4배열의 1번째 값을 읽어오기

```
document.write(arr4[0], "<br>");//[1,2,3]
```

//arr4배열의 2번째 값을 읽어오기

```
document.write(arr4[1], "<br>");//[4,5,6]
```

//3값을 읽어와서 찍어보자.

//arr4[0]이 배열이므로 이배열에 []를 사용해서 원하는 값을 찾는다.

```
document.write(arr4[0][2], "<br>");//3
```

//12값을 읽어와서 찍어보자.

```
document.write(arr4[3][2], "<br>");
```

//10값을 찾아서 100으로 변경해 보자.

```
arr4[3][0]=100;
```

```
document.write(arr4, "<br>");
```

//2차원 배열 출력하는 방법

```
for (var i = 0; i < arr4.length; i++) { //arr4.length 4
    for (var j = 0; j < arr4[0].length; j++) { //arr4[0].length 3
```

```

        console.log(arr4[i][j]); // 2차원 배열의 모든 내용 출력
    }
}

```

//2차원 배열 복사본 만들기

```

var arr = [];
for (var i = 0; i < arr4.length; i++) { //arr4.length 4
    arr[i] = [];
    for (var j = 0; j < arr4[0].length; j++) { //arr4[0].length 3
        arr[i][j] = arr4[i][j]; // 초기화 및 원하는 데이터로 채울 수 있다.
    }
}

```

```

<!DOCTYPE html>

```

```

<html lang="en">

```

```

<head>

```

```

    <meta charset="UTF-8">

```

```

    <title>Document</title>

```

```

    <script>

```

```

var arr=[3,6,1,2,4];    //배열 생성 방법

```

```

document.write(arr,"<br>");    //배열의 내용출력

```

```

document.write(arr[0],"<br>");    //배열 읽는 방법

```

```

arr[0]=20;                //배열에 값을 넣는 방법

```

```

document.write(arr,"<br>");

```

```

arr[10]=50; //범위에 벗어나면 알아서 생성한다. 사이 공간의 빈체로 만들어 준다.

```

```

document.write(arr,"<br>");

```

```

var arr2;

```

```

arr[10]=10;    //정상

```

```

//arr2[10]=20;    //error arr2는 배열이 아니여서 [] 연산자를 사용할 수 없다.

```

```

arr2=[];        //배열로 만들어야 문제없이 사용할수 있다.

```

```

arr2[10]=20;

```

```

var arr3=new Array();//변수를 배열 객체로 선언

```

```

//arr배열을 가지고 다음을 풀어보자.

```

```

//배열안에 모든 값을 더한 합을 구해보자.

```

```

var sum=0;
for(var i=0;i<arr.length;i++){
    sum=sum+arr[i];
}
document.write(sum,"<br>");
//in 배열에 있는 내용을 하나씩 가져와서 실행한다.
var sum=0;
for(var i in arr){
    sum=sum+i;
}
document.write(sum,"<br>");
//배열안에 기존의 값에 3을 곱한 값을 넣어보자.
for(var i=0;i<arr.length;i++){
    arr[i]=arr[i]*3;
}
document.write(arr,"<br>");
//배열안에 모든 값을 더한 합을 구해보자.
//배열안에 기존의 값에 3을 곱한 값을 넣어보자.
//.length 배열에 길이를 생성해 준다.
//.concat() 원소 또는 배열을 합친다.
var arr1=[1,2,3,4];
var arr2=arr1.concat(5);
document.write(arr1,"<br>");//1,2,3,4
document.write(arr2,"<br>");//1,2,3,4,5
var arr3=[2,3,4].concat([5,6,7]);//두배열을 합친다.
document.write(arr3,"<br>");//1,2,3,4,5

//.every() 배열의 데이터가 모두 조건에 맞는지 확인
// 실행 결과 true,false 값을 가진다.
//콜백함수의 결과가 모두 true이면 true, 하나라도 만족하지 않으면 false
var arr1=[1,2,3,4];
var result=arr1.every(function(x){
    return x<5;
})
document.write(result,"<br>");//true

```

```

var result=arr1.every(function(x){
    return x<2;
})
document.write(result,"<br>");//false
//filter 배열에서 조건에 맞는 값만 추출해서 새로운 배열을 만듦
var arr1=[1,2,3,4,5];
var result=arr1.filter(function(x){
    return x%2===1;
});
document.write(arr1,"<br>");//false//1,2,3,4,5
document.write(result,"<br>");//1,3,5

```

```

var f1=function(x){
    return x%2===1;
};
var result=arr1.filter(f1);

```

//화살표함수

/*화살표 함수는 JavaScript에서 함수를 간단하게 만드는 방법입니다. 기존의 함수를 좀 더 짧고 간결하게 작성할 수 있게 도와줍니다. 두 방식의 결과의 차이점은 없습니다.

기존 함수:

```

function greet(name) {
    return "안녕하세요, " + name + "님!";
}

```

화살표 함수:

```

var greet = name => "안녕하세요, " + name + "님!";

```

다음은 화살표 함수 설명하는 내용입니다.

함수 선언: function 키워드 대신 => 기호를 사용하여 함수를 선언합니다.

인자(매개변수): 화살표 앞쪽에 인자가 들어 간다. 인자가 없으면(), 여러 개의 인자가 있을 때는 괄호로 감싸줍니다. 예를 들어 (x, y)=>, 하나이면 중괄호를 생략할 수 있다. 예를 들어, x =>

함수 본문: 함수 본문은 화살표(=>) 다음에 나옵니다. 본문이 한줄일 경우 중괄호 {}로 감싸지 않아도 됩니다.

리턴 값: 본문이 한줄일 경우 별도의 return 키워드를 사용하지 않아도 됩니다.

간단한 예제로 화살표 함수를 설명하겠습니다:

```
// 기존 함수
```

```
function add(x, y) {  
    return x + y;  
}
```

```
// 화살표 함수로 변환
```

```
var add = (x, y) => x + y;
```

위 예제에서, add 함수를 화살표 함수로 변환하면 더 간결한 코드가 됩니다. 화살표 함수는 주로 익명 함수로 사용되며, 함수를 변수에 할당하거나 배열의 map, filter, reduce와 같은 고차 함수와 함께 사용되어 코드를 더 읽기 쉽게 만들어줍니다. */

```
var result=arr1.filter(x=>x%2===1);  
//(x,y)=>x+y;      //다음 3개는 같은 의미이다.  
//(x,y)=>{x=x+y;return x;};  
  
//.indexOf      앞에서 부터 원하는 데이터의 인덱스를 찾는다.  
//.lastIndexOf  뒤에서 부터 찾기  
var arr1=[1,2,3,4];  
document.write(arr1.indexOf(2), "<br>");//1  
document.write(arr1.lastIndexOf(3), "<br>");//2  
//indexOf(찾을 데이터, 시작 인덱스)  
document.write(arr1.indexOf(1,2), "<br>");//못찾으면 -1  
  
//.join 배열을 합쳐서 하나의 문자열을 만들  
var arr1=[1,2,3,4];  
document.write(arr1.join("-"), "<br>");//1-2-3-4  
  
//자료구조 데이터를 사용하는 방법  
//변수 ,배열 , list ,stack, que,트리, 그래프 등이 자료구조에 해당한다.  
//변수 하나의 데이터저장  
//배열 여러개의 데이터저장  
//list 연속된 데이터  
// 스택 LIFO last in first out 마지막 들어간 것이 먼저 나옴  
// 큐 FIFO first in first out 처음들어간 데이터가 먼저 나옴  
// 두 메소드를 사용해서 배열을 stack처럼 사용할 수 있다.
```

스택 (Stack)

후입선출(LIFO, Last-In-First-Out) 구조를 가지고 있습니다.

새로운 항목은 항상 스택의 맨 위에 추가되고, 제거할 때도 맨 위에서부터 제거됩니다.

주로 함수 호출, 실행 컨텍스트 관리, 뒤로 가기 기능 등에 사용됩니다.

기본 동작:

push: 스택에 항목을 추가합니다.

pop: 스택의 맨 위 항목을 제거합니다.

peek 또는 top: 스택의 맨 위 항목을 조회합니다.

```
// .push(), pop() 메소드 push데이터 삽입 , pop 데이터를 읽어와 삭제
var arr1=[1,2,3,4];
arr1.push(5);
document.write(arr1,"<br>");//1,2,3,4,5
arr1.push(6);
document.write(arr1,"<br>");//1,2,3,4,5,6
document.write(arr1.pop(),"<br>");//6
document.write(arr1,"<br>");//1,2,3,4,5
document.write(arr1.pop(),"<br>");//5
document.write(arr1,"<br>");//1,2,3,4
//큐 que
```

큐 (Queue)

선입선출(FIFO, First-In-First-Out) 구조를 가지고 있습니다.

새로운 항목은 항상 큐의 뒤쪽에 추가되고, 제거할 때는 앞쪽에서부터 제거됩니다.

주로 작업 대기열, 이벤트 처리 등에 사용됩니다.

기본 동작:

enqueue: 큐의 뒤쪽에 항목을 추가합니다. push() 배열의 마지막에 값을 추가한다.

dequeue: 큐의 앞쪽 항목을 제거합니다. shift() 배열의 첫번째 값을 읽어와 제거한다.

front 또는 peek: 큐의 앞쪽 항목을 조회합니다.

```
var queue = [1, 2, 3, 4];
// 요소를 큐 뒤쪽에 추가
queue.push(5);
// 큐의 맨 앞 요소를 제거
var removedElement = queue.shift();
console.log(removedElement); // 1
console.log(queue); // [2, 3, 4, 5]
```

```

//.sort
var arr1=[5,1,3,2,4];
document.write(arr1.sort(),"<br>");
document.write(arr1,"<br>");//1,2,3,4,5
var arr1=['b','c','a'];
document.write(arr1.sort(),"<br>");
document.write(arr1,"<br>");//a,b,c
//sort는 기본적으로 문자열 정렬되어 다음과 같이 출력된다.
var arr1=[5,1,3,2,4,10];
document.write(arr1.sort(),"<br>");//1,10,2,3,4,5
document.write(arr1,"<br>");//1,10,2,3,4,5 arr1자체가 정렬된다.
//sort를 숫자로 정렬 하고 싶으면 콜백함수를 사용해야 한다.
//매개변수가 함수인경우 콜백함수라 한다.
arr1.sort(function(a,b){//정렬방식을 재정의해서 원하는 정렬을 할 수 있다.
    //오름차순정렬 a,b비교결과를 1,0,-1로 적절히 조절해서 정렬 할 수 있다.
    if(a>b)return 1;
    if(a<b)return -1;
    if(a==b)return 0;
    //내림차순정렬
    //if(a>b)return -1;
    //if(a<b)return 1;
    //if(a==b)return 0;
});
document.write(arr1,"<br>");
//reverse 배열의 내용을 역정렬한다.
arr1.reverse();
document.write(arr1,"<br>");
//.slice 배열에 특정 부분을 추출한다.
var arr1=[5,1,3,2,4,10];
//인덱스 3이후 모두 출력
document.write(arr1.slice(3),"<br>");//2 4 10
//인덱스 2부터 인덱스 5 앞쪽의 데이터까지
document.write(arr1.slice(2,5),"<br>");//3 2 4

```

```

document.write(arr1, "<br>");//변경되지 않은

//.splice 추출 및 변경한다.
//2번 인덱스부터 2개 추출
//리턴값은 삭제된 데이터가 리턴된다.
var arr1=[5,1,3,2,4,10];
var remove=arr1.splice(2,2);
document.write(arr1, "<br>");//5,1,4,10
document.write(remove, "<br>");//3,2

var arr1=[5,1,3,2,4,10];
var remove=arr1.splice(2,2,999,998);
document.write(arr1, "<br>");//5,1,999,998,4,10
document.write(remove, "<br>");//3,2

var arr4=[
    [1,2,3],
    [4.5,6],
    [7,8,9],
    [10,11,12]
]
//2중 for문을 이용해서 배열의 내용을 출력해 보자.
//2차원 배열에 들어 있는 값을 html table로 만들어서 해당 td에 넣어보자.
for(var i=0;i<4;i++){
    for(var j=0;j<3;j++){
        document.write(arr4[i][j], " ")
    }
    document.write("<br> ")
}
document.write("<table border=1>")
for(var i=0;i<4;i++){
    document.write("<tr>")
    for(var j=0;j<3;j++){
        document.write("<td>")
        document.write(arr4[i][j])
    }
}

```



```
        document.write("<td>")
    }
    document.write("<tr>")
}
document.write("</table>")
</script>
</head>
<body>
</body>
</html>
```

> 12. 자바스크립트 객체 json

json이란? 자바스크립트 객체를 의미한다. JSON (JavaScript Object Notation)은 데이터를 저장하고 교환하는 용도로 사용된다. 클래스 변수(인스턴스)에 .를 찍어 값에 접근한다.

```
var a={}; // a는 객체가 된다.
```

```
var b={
    name: "John Doe",
    age: 30,
    isStudent: false,
    hobbies: ["reading", "swimming"],
    address: {
        street: "123 Main St",
        city: "Anytown",
```

```

    zipCode: "12345"
  }
}

```

a.test=20; b.a=10; //선언후 동적 할당이 가능하다. 각 객체에 동적할당한 예제

객체 (Object): 중괄호 {}로 표현하며, 키와 값 사이에 콜론(:)으로 구분합니다. 여러 개의 키-값 쌍을 가질 수 있으며, 각 쌍은 쉼표로 구분됩니다. 키는 문자열이어야 하며, 값은 다양한 데이터 타입을 가질 수 있습니다.

JSON 문법에서 주의할 점:

키(key)는 반드시 문자열이어야 하며, 쌍 따옴표로 감싸야 합니다. 쌍 따옴표는 큰 따옴표(")나 작은 따옴표(') 모두 사용 가능 하고, 특별한 경우를 제외하고 생략할 수 있다.

값의 데이터 타입으로는 문자열, 숫자, 불리언, 배열, 객체, null이 허용됩니다.

JSON은 데이터를 표현하는 데 간단하고 일관성이 있으며, 다양한 프로그래밍 언어에서 지원되기 때문에 데이터 교환과 저장에 매우 효과적입니다.

JSON.parse()와 JSON.stringify() 함수를 이용하여 문자열을 json으로 json를 문자열로 형변환할 수 있다.

객체 속성을 제거하기 위해서 delete를 사용한다. `delete(human.email);`

```

// JSON 문자열 파싱
var jsonString = '{"name": "John Doe", "age": 30}';
var jsonObject = JSON.parse(jsonString);
console.log(jsonObject.name); // "John Doe"
// JavaScript 객체를 JSON 문자열로 변환
var person = { name: "Alice", age: 25 };
var jsonPerson = JSON.stringify(person);
console.log(jsonPerson); // '{"name":"Alice","age":25}'

```

이러한 함수를 사용하여 JavaScript 객체와 JSON 문자열 간의 변환을 수행할 수 있습니다.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script>
    var obj={};

```

```
document.write(typeof obj, "<br>")
```

```
var human={  
  name: '홍길동',  
  age: 32,  
  friends: ['고길동', '고길남'],  
  address: {  
    id: 234,  
    area: '천안',  
    comment: '주소임'  
  },  
  myFunction: function(){  
    alert(this.name);  
  }  
}
```

```
document.write(human.name, "<br>")
```

```
human.name = "홍길남"
```

```
document.write(human.name, "<br>")
```

```
document.write(human.address.area, "<br>")
```

```
human.myFunction();
```

//객체에 동적으로 새로운 필드를 추가하려면 기존 객체에 .를 찍어서 새로운 필드명으로 값을 넣을 수 있다.

```
human.email = 'hello@daum.net';
```

```
human.email = 'hello@daum.net'; //에러가 아니다.
```

```
console.log(human);
```

```
delete(human.email); //속성제거
```

```
console.log(human);
```

```
</script>
```

```
</head>
```

```
<body>
```

```
</body>
```

```
</html>
```

> 13. 이벤트

이벤트란?

상태변화가 일어 났을때 발생하는 신호를 이벤트라 한다.

웹에서 발생할 수 있는 대표적인 이벤트로 마우스(클릭,드래그,더블클릭), 키보드(특정 키 누르기), 웹페이지에서 상태의 변화(특정 객체 선택, 화면이 다 불러와진 다음)가 있다.

이벤트 핸들러란? 이벤트가 발생할 때 실행되는 특정 함수를 의미한다.

이벤트 종류는 다음과 같다.

load 화면이 모두 로딩된 다음에 실행되는 이벤트

focus 해당 객체가 선택되었을때 발생하는 이벤트

blur 해당 객체가 선택을 잃어버렸을때 발생하는 이벤트

Click 클릭하였을때 발생

dblclick 연속 두번 클릭하였을때 발생

mouseover 마우스가 해당 객체 위에 있을때 발생

mouseenter 마우스가 들어 왔을 때

mouseleave 마우스가 떠났을때

이벤트에 따라서 연속키나 특수키 조작이 안되니 웹에서 찾아서

상황에 맞춰서 적절한 이벤트를 선택해 보자.

keydown 키를 눌렀을때

keyup 키를 놓았을때

keypress 키를 눌렀을 때

이벤트를 적용하는 방법은 html 속성에서 사용하는 방법과 javascript 객체에서 사용하는 방법 2가지가 있다.

HTML 속성으로 이벤트 실행하는 방법은 다음과 같은데 되도록 사용하지 말고 javascript를 사용할 것을 권장한다.

HTML 속성을 사용하여 이벤트를 처리하는 방법:

<요소명 이벤트속성="JavaScript 코드">

이것은 HTML 요소의 속성으로 이벤트 핸들러 함수를 직접 지정하는 방식입니다. 이벤트 속성은 이벤트 종류중 하나 선택해서 이벤트 이름에 on를 추가하면 된다. 예를 들면 click이벤트는 onClick, Load이벤트는 onLoad와 같이 기술 하면 된다. 이벤트 속성이 선택되면 javascript 코드 부분에 이벤트 핸들러로 사용할 함수를 추가하면 해당 이벤트가 발생 할 때 이벤트 핸들러로 지정한 함수가 실행 된다.

body태그안에 다음을 기술하고 버튼을 클릭해 보자.

<button onclick="myFunction()">클릭하세요</button>

<script>

```
function myFunction() {  
    // 이벤트 처리 동작을 정의하는 코드  
    alert('버튼이 클릭되었습니다.');
```

```
}
```

</script>

여기서 onclick은 클릭 이벤트를 처리하는 속성이며, myFunction()은 클릭 이벤트가 발생했을 때 실행될 JavaScript 함수입니다.

이 경우 myFunction() 함수는 이벤트 핸들러이고, 버튼이 클릭되면 경고 창을 띄운다.

인자 전달: 이벤트 핸들러 함수에는 일반적으로 이벤트 정보를 전달할 수 있습니다. 이를 통해 함수 내에서 이벤트 추가 정보를 활용할 수 있습니다. 아래는 클릭 이벤트에서 이벤트 정보를 전달하는 예시입니다.

```
<button onclick="myFunction(event)">클릭하세요</button>
```

```
script>
```

```
function myFunction(event) {  
    // event 매개변수 객체를 이용해 다양한 추가 정보를 활용할 수 있다.  
    alert('클릭 위치: ' + event.clientX + ', ' + event.clientY);  
}
```

```
</script>
```

이와 같이 HTML 속성을 사용하여 인라인 이벤트 핸들러를 정의하면 간단한 이벤트 처리에 용이하지만, 복잡한 이벤트 처리 및 코드 유지보수에는 제한적일 수 있습니다. 따라서 대규모 프로젝트나 더 모듈화된 코드를 작성할 때는 외부 JavaScript 파일을 활용하는 것이 일반적이다.

다음 예제를 기술해 보고 기술된 이벤트들을 확인해 보자.

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <title>Document</title>
```

```
    <script></script>
```

```
</head>
```

```
<body onload="alert('화면이 다 불러지고나면 메시지 발생')" onresize="alert('브라우저 크기를 변경하였으면 발생')">
```

```
    <input type="text" onfocus="alert('선택되면 포커스 이벤트가 발생한다.')">
```

```
    <input type="text" onblur="alert('포커스를 잃어버리면 발생 한다.')">
```

```
    <div onclick="alert('onclick')">
```

마우스 이벤트 글씨를 클릭하면 onclick 메시지박스가 생성된다.

```
</div>
```

```
    <textarea
```

```
onkeypress="alert(event.keyCode+'keyPress:'+String.fromCharCode(event.keyCode))
```

```
"></textarea>
```

```
</body>
```

```
</html>
```

onkeypress는 키보드가 눌려졌을 때 발생 하는 이벤트이다.

event.keyCode는 이벤트가 발생했을때 누른 문자 코드 숫자 값이 저장된다.

event객체는 이벤트의 추가정보를 담는 미리 선언된 약속된 객체라고 생각하면 된다.

String.fromCharCode()는 사람이 알아볼 수 있도록 문자 코드 숫자를 실제 문자로 변경하는 함수이다.

> 14. DOM을 이용한 이벤트

DOM(Document Object Model)객체는 HTML 문서의 구조와 내용을 조작하고 다루는 데 사용됩니다. 원하는 HTML태그를 DOM으로 찾은 다음 dom를 통해서 해당 html태그에 이벤트를 추가한다. document객체가 dom에 해당 한다고 생각해도 크게 다르지 않다.

다음 코드는 "button1"이라는 id를 가진 HTML 요소를 찾고, 그 요소에 클릭 이벤트 핸들러를 할당하여 버튼이 클릭될 때 경고 창을 표시하는 동작을 정의하는 것입니다.

```
document.getElementById('button1').onclick = function() {  
    alert('버튼이 클릭되었습니다!');  
};
```

위의 코드는 JavaScript에서 DOM 요소의 이벤트 핸들러를 설정하는 방법을 보여주고 있습니다. 아래는 그 코드의 구문을 자세히 설명하는 내용입니다:

document.getElementById('button1'):

document.getElementById는 문서 객체 모델(DOM)에서 HTML 요소를 찾기 위한 함수입니다.

'button1'은 HTML 요소의 id 속성을 기반으로 찾고자 하는 요소를 지칭하는 문자열입니다.

이 함수는 해당 id를 가진 요소를 반환합니다.

.onclick:

onclick은 HTML 요소의 클릭 이벤트에 대한 이벤트 핸들러 속성입니다.

이 속성에 함수를 할당하면 해당 요소가 클릭될 때 그 함수가 실행됩니다.

```
function() { alert('버튼이 클릭되었습니다!'); }:
```

이 부분은 익명 함수(anonymous function)로 이벤트 핸들러 구현함. 함수 내부에 alert 함수를 사용하여 경고 창을 표시하고 있으며, 메시지로 "버튼이 클릭되었습니다!"를 출력합니다.

이벤트 구현하는 방법은 결국 찾아서 이벤트 추가하는 것이 대부분이어서 찾는 방법과 이벤트 추가 방법을 공부하면 된다.

대상을 찾는 방법은 다음과 같다.

1. `getElementById(id)`:

```
<html>

  <body>

    <div id="myElement">안녕하세요!</div>

  </body>

</html>
```

이 예제에서는 `getElementById` 함수를 사용하여 id가 "myElement"인 요소를 가져오는 자바스크립트는 다음과 같다.

```
var element = document.getElementById("myElement");

console.log(element); // <div id="myElement">안녕하세요!</div>
```

2. `getElementsByClassName(className)`:

```
<html>

  <body>

    <p class="myClass">첫 번째 문단</p>

    <p class="myClass">두 번째 문단</p>

  </body>

</html>
```

이 예제에서는 `getElementsByClassName` 함수를 사용하여 클래스 이름이 "myClass"인 모든 요소를 가져오는 자바스크립트는 다음과 같다.

```
const elements = document.getElementsByClassName("myClass");

for (var element of elements) {

  console.log(element); // "첫 번째 문단" // "두 번째 문단"
```



```
}
```

3. `getElementsByName(tagName)`:

```
<html>
```

```
  <body>
```

```
    <p>첫 번째 문단</p>
```

```
    <p>두 번째 문단</p>
```

```
  </body>
```

```
</html>
```

이 예제에서는 `getElementsByName` 함수를 사용하여 모든 `<p>` 태그를 가진 요소를 가져오는 자바스크립트는 다음과 같다.

```
var paragraphs = document.getElementsByTagName("p");
for (var paragraph of paragraphs) {
  console.log(paragraph); // "첫 번째 문단" // "두 번째 문단"
}
```

4. `querySelector(selector)`:

```
<html>
```

```
  <body>
```

```
    <div class="myClass">첫 번째 div</div>
```

```
    <div class="myClass">두 번째 div</div>
```

```
  </body>
```

```
</html>
```

이 예제에서는 `querySelector` 함수를 사용하여 CSS 선택자 `.myClass`에 일치하는 첫 번째 요소를 가져오는 자바스크립트는 다음과 같다.

```
const element = document.querySelector(".myClass");
console.log(element); // "첫 번째 div"
```

5. `querySelectorAll(selector)`:

```

<html>

  <body>

    <div class="myClass">첫 번째 div</div>

    <div class="myClass">두 번째 div</div>

  </body>

</html>

```

이 예제에서는 `querySelectorAll` 함수를 사용하여 CSS 선택자 `.myClass`에 일치하는 모든 요소를 가져오는 자바스크립트는 다음과 같다.

```

const elements = document.querySelectorAll(".myClass");

elements.forEach((element) => {

  console.log(element); // "첫 번째 div" // "두 번째 div"

});

```

상위에서 찾은 `element`에 `.onclick`와 핸들러를 추가하면 클릭 이벤트를 구현할 수 있다. `.onclick`부분에 이전에 배운 다른 이벤트 이름으로 바꾸면 해당 이벤트를 구현할 수 있다. `on`다음에 이벤트이름을 기술하면 된다.

원하는 태그에 이벤트를 추가하고 싶다면 상위 코드를 이용해 원하는 태그를 찾은 다음 원하는 이벤트를 추가하면 된다.

화면이 다 불러지고 실행해야 할 코드가 있는 경우 다음과 같이 `window.onload`에 이벤트 핸들러를 추가하면 된다. 보통 `html`코드에서 `DOM`를 이용해서 태그를 찾을때 화면이 다 불러지지 않은 상태에서 실행되면 원하는 태그를 찾지 못한다. `DOM`를 사용해서 태그를 찾으려면 `onload`이벤트를 이용해서 화면이 다 불러지고 나서 실행되도록 해야 원하는 태그를 찾을 수 있다.

`javascript`를 `head`태그에 기술하면 `body`에 기술된 `html`태그를 찾지 못한다. `javascript` 실행 시점에 `html`태그를 읽지 못해서 그렇다. 이럴 때 `onload`이벤트를 이용해서 화면이 다 불러와 진 다음에 `javascript`코드를 실행하게 할 수 있다. 여기서 `window` 객체는 자바스크립트 최상위 전역 객체라고 생각하면 된다. `window.onload`는 화면이 다 불러진 다음에 실행하라는 이벤트라 생각 하면 된다.

```

window.onload=function(){

  //코드

}

```

해더에 기술해도 문제 없이 프로그램을 찾을 수 있다.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">

```

```

    <title>이벤트 예제</title>
    <script>
window.onload = function() {
    // 버튼 요소를 가져옵니다.
    var myButton = document.getElementById("myButton");
    // 버튼에 클릭 이벤트 핸들러를 추가합니다.
    myButton.onclick = function() {
        alert("버튼이 클릭되었습니다!");
    };
};
</script>
</head>
<body>
    <button id="myButton">클릭하세요</button>
</body>
</html>

```

javascript Dom객체로 다음과 같은 방법으로 HTML를 직접 접근할 수 있다.

document.images.img2.src 이 코드는 현재 웹 페이지에서 id가 "img2"인 HTML 이미지 요소의 src속성을 가리킨다.

JavaScript에서 document.images 객체는 현 화면의 모든 이미지를 가지고 있는 객체라 생각하면 된다.

document.images.img2.src 은 images 객체 내에서 id 속성이 "img2"인 이미지 요소의 src 속성에 접근함을 의미 한다.

document: 현재 웹 페이지의 문서 객체 모델(Document Object Model, DOM)에 접근하는 JavaScript의 내장 객체입니다.

images: document 객체의 하위 속성으로, 현재 웹 페이지에 있는 모든 이미지 요소를 포함하는 컬렉션(배열과 유사한 구조)입니다.

img2: name 또는 id가 "img2"인 이미지 요소를 찾는 데 사용되는 식별자입니다.

src: 이미지 요소의 속성 중 하나로, 이미지 파일의 소스 경로(URL)를 나타냅니다.

이 코드가 동작하려면 id가 "img2"인 이미지 요소가 웹 페이지에 존재해야 합니다. 아래 예제에서 id="img2" 부분이 선택 된다.

```

<body>

</body>

```

다음 3가지 방식은 같은 결과를 가진다.

document.images.img2.src="new_image.gif"; // 찾은 이미지 태그의 이미지를 변경

```
document.getElementById("img2").src = "new_image.gif";
```

```
document.images[1].src = "new_image.gif";
```

document.images[1]은 화면의 두 번째 이미지 요소를 가리키고, 그 후 .src를 통해 src 속성을 변경할 수 있습니다. 이미지가 화면에 많이 존재 하면 원하는 이미지 찾기가 어려워 잘 사용하지 않는다.

다음 예제는 "img2"라는 id를 가진 이미지 태그를 마우스 오버 시 "2.gif"로 바꿔주고 마우스 아웃 시 "1.gif"로 바꿔 준다. 웹에서 적절한 이미지를 다운로드 해서 다음 예제를 실행 해 보자.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    window.onload=function(){//화면이 다 호출된 후 다음 코드가 실행된다.
//화면이 다 불러지지 않고 다음 코드가 실행 되면 위에서 아래로 실행되는 코드 특성상 아직
생성되지 않은 html 구조 DOM를 호출 해서 오류가 발생 된다. 이런 문제를 해결 하기 위해서
onload 이벤트에 Dom조작 관련 코드를 등록 하였다.
```

다음예제는 마우스를 이미지 위에 올리면 이미지가 변경되고, 마우스를 이미지에서 떼면 다시 원래 이미지로 돌아가는 예제이다.

```
    var img2=document.getElementById("img2");
    img2.onmouseover=function(){
      //document.images.img2.src='2.gif'
      img2.src='2.gif';
    }
    img2.onmouseout=function(){
      //document.images.img2.src='1.gif'
      img2.src='1.gif';
    }
  }
</script>
</head>
<body>
JavaScript를 사용하여 이미지를 변경하는 방식이 아닌 HTML 속성을 사용하여 이미지를 변경
방식<br>
  <a href="http://www.daum.net">
    
  </a>
  <a href="http://www.daum.net">
    
  </a>
```

```

</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    /*
      document.getElementById("id");      //id이용해서 찾기
      document.getElementsByTagName("div");//태그를 이용해서 찾기
      document.getElementsByClassName("myClass");//class이용해서 찾기
      document.querySelectorAll();        //css선택자를 이용해서 찾기
    */
    window.onload=function(){
      var myP = document.getElementsByTagName("P");//배열로 생성됨
      //alert(myP[0].innerHTML);//p1태그1
      //alert(myP[1].innerHTML);
      var myP = document.getElementsByClassName("myClass");
      //alert(myP[0].innerHTML);//p1태그2
      //alert(myP[1].innerHTML);
      var myP = document.querySelectorAll("p.myClass");
      //alert(myP[0].innerHTML);//p1태그2
      //alert(myP[1].innerHTML);
      var myP = document.querySelectorAll(".myClass2");
      alert(myP[0].innerHTML);//p1태그1
    }
  </script>
</head>
<body>
  <p class="myClass2">p1태그1</p>
  <p class="myClass">p1태그2</p>
  <p class="myClass">p1태그3</p>
</body>
</html>

```

> 14. dom에서 내용 읽어오기

사용자 입력을 처리하는 3가지 방법

innerText로 텍스트를 가져옴.

innerHTML로 HTML을 가져옴.

value로 <input> 요소의 사용자 입력값을 가져옴.

다음은 읽어오는 예제이다.

```
<!DOCTYPE html>
<html>
<head>
  <title>innerText vs. innerHTML vs. value 차이 예제</title>
</head>
<body>
  <!-- innerText 예제 -->
  <div id="innerTextExample">
    이것은 <strong class="hide">innerText</strong> 예제입니다.
  </div>
  <!-- innerHTML 예제 -->
  <div id="innerHTMLExample">
    이것은 <strong class="hide">innerHTML</strong> 예제입니다.
  </div>
  <!-- value 예제 -->
  <input type="text" id="valueExample" value="이것은 value 예제입니다." />
</script>
```

```

// innerText로 텍스트 가져오기
var innerTextElement = document.getElementById("innerTextExample");
var innerTextValue = innerTextElement.innerText;
// innerHTML로 HTML 가져오기
var innerHTMLElement = document.getElementById("innerHTMLExample");
var innerHTMLValue = innerHTMLElement.innerHTML;
// input 요소의 value 가져오기
var valueInputElement = document.getElementById("valueExample");
var valueInputValue = valueInputElement.value;
console.log("innerText 값: " + innerTextValue);
console.log("innerHTML 값: " + innerHTMLValue);
console.log("value 값: " + valueInputValue);
</script>
</body>
</html>

```

다음은 변경하는 예제이다.

```

<!DOCTYPE html>
<html>
<head>
  <title>innerText vs. innerHTML vs. value 변경 예제</title>
</head>
<body>
  <!-- innerText 변경 예제 -->
  <div id="innerTextExample">
    이것은 <strong class="hide">innerText</strong> 예제입니다.
  </div>
  <!-- innerHTML 변경 예제 -->
  <div id="innerHTMLExample">
    이것은 <strong class="hide">innerHTML</strong> 예제입니다.
  </div>
  <!-- value 변경 예제 -->
  <input type="text" id="valueExample" value="이것은 value 예제입니다." />
  <script>
    // innerText로 텍스트 변경하기
    var innerTextElement = document.getElementById("innerTextExample");
    innerTextElement.innerText = "innerText 값이 변경되었습니다.";
    // innerHTML로 HTML 변경하기
    var innerHTMLElement = document.getElementById("innerHTMLExample");
    innerHTMLElement.innerHTML = "innerHTML 값이 <em>변경</em>되었습니다.";
    // input 요소의 value 변경하기
    var valueInputElement = document.getElementById("valueExample");
    valueInputElement.value = "value 값이 변경되었습니다.";
    // 변경된 값 콘솔에 출력
    console.log("innerText 값: " + innerTextElement.innerText);
  </script>

```

```

        console.log("innerHTML 값: " + innerHTML);
        console.log("value 값: " + valueInputElement.value);
    </script>
</body>
</html>

```

innerHTML를 사용해서 원하는 html태그 추가하기

다음은 버튼을 누르면 html구조가 변경되는 코드이다.

```

<!DOCTYPE html>
<html>
<head>
    <title>HTML 테이블 추가 예제 (createElement와 appendChild 없이)</title>
    <script>
        function addTable() {
            // onLoad에 정의 하지 않은 이유는 함수는 화면이 다불려진 후에 사용되기 때문이다.
            // 테이블을 만들기 위해 HTML 문자열을 생성
            var tableHTML = `
                <table>
                    <tr>
                        <td>셀 1-1</td><td>셀 1-2</td>
                    </tr>
                    <tr>
                        <td>셀 2-1</td><td>셀 2-2</td>
                    </tr>
                </table>
            `;
            // tableContainer에 새로운 테이블의 HTML 문자열을 추가
            var tableContainer = document.getElementById("tableContainer");
            tableContainer.innerHTML += tableHTML;
            // 추가된 HTML을 출력
            var addedHTML = tableContainer.innerHTML;
            console.log("추가된 HTML:\n" + addedHTML);
        }
    </script>
</head>
<body>
    <h1>테이블 추가 예제</h1>
    <button onclick="addTable()">테이블 추가</button>
    <div id="tableContainer">
        <!-- 새로운 테이블이 여기에 추가될 것입니다. -->
    </div>
</body>
</html>

```



상단의 버튼모양 이미지를 클릭하면
클릭한 주제로 내용이 변경되는

프로그램을 구현해 보자.

> 16. addEventListener 사용한 이벤트 처리

이전에 사용한 많은 이벤트 방식이 있는데 사용할 것을 권하지 않고 `addEventListener`를 사용할 것을 권한다.

```
element.addEventListener(event, handlerFunction);
```

`element`: 이벤트를 연결할 HTML 요소를 나타냅니다.

`event`: 연결하려는 이벤트의 종류를 나타내는 문자열입니다 (예: "click", "mouseover", "keydown" 등).

`handlerFunction`: 이벤트가 발생했을 때 실행할 함수 또는 콜백 함수입니다.

사용예제는 다음과 같다.

```
// HTML에서 버튼 요소를 가져오기
```

```
var button = document.getElementById("myButton");
```

```
// 버튼에 클릭 이벤트 핸들러 추가
```

```
button.addEventListener("click", function() {
```

```
    alert("버튼이 클릭되었습니다!");
```

```
});
```

이벤트에 사용한 이벤트 핸들러 함수를 이용해서 추가한 이벤트를 삭제할 수 있다.

이벤트 삭제하기 `removeEventListener`

```
const button = document.getElementById("myButton");
```

```
function handleClick() {
```

```
    alert("버튼이 클릭되었습니다!");
```

```
}
```

```
button.addEventListener("click", handleClick);
```

```
// 이벤트 핸들러 삭제
```

```
button.removeEventListener("click", handleClick);
```

삭제할 이벤트 핸들러를 이벤트와 함께 `removeEventListener`함수의 매개변수에 기술 하면된다.

이벤트 추가정보 확인하기 핸들러 함수의 매개변수에 다양한 이벤트 정보를 얻을 수 있다.

```
const button = document.getElementById("myButton");
button.addEventListener("click", function(event) {
    console.log("클릭된 요소: " + event.target.tagName);
    console.log("마우스 위치: " + event.clientX + ", " + event.clientY);
    if (event.target.tagName === "BUTTON") {// 버튼이 클릭된 경우 만 처리
        alert("버튼이 클릭되었습니다!");
    }
});
```

`addEventListener`는 여러개의 핸들러 함수를 부분적으로 등록 삭제 할 수 있다.

다음 예제는 버튼에 `f1`, `f2`함수를 핸들러로 등록한다음 `f2`함수만 삭제한 예제이다.

2개 함수를 등록하면 버튼을 클릭하면 2개다 실행이되고 1개를 삭제하면 나머지 한개만 남는다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script>
        function f1(){
            alert("f1");
        }
        window.onload=function(){
            //onClick 같은 경우 이벤트 등록이 1개만된다.
            //addEventListener의 경우 클릭이벤트를 여러개 등록할 수있다.
            //다음 코드는 2개 클릭이벤트를 등록하고 하나의 이벤트만 삭제하였다.
            var b1=document.getElementById("b1");
            b1.addEventListener('click',f1);
            b1.addEventListener('click',function(){
                alert("f2");
            });
            b1.removeEventListener('click',f1);
        }
    </script>
</head>
<body>
    <button id="b1">클릭</button>

</body>
</html>
```

> 17. 자판기 만들기

자 판 기			
 1200₩	 1100₩	 1000₩	1000원 500원 100원 0원
 400₩	 400₩	 400₩	잔돈 반환
			500원 1개 100원 2개

다음 자판기 사용자 매뉴얼보고 자판기를 만들어 보자.

1. 시작하기

웹 브라우저를 열고 index.html 파일을 더블 클릭하여 자판기 프로그램을 시작합니다.

자판기 화면이 표시됩니다. 화면 상단에는 "자판기"라는 제목이 표시되어 있습니다.

2. 금액 투입하기

자판기에 금액을 투입할 수 있습니다. 투입할 금액 버튼은 오른쪽 하단에 있습니다.

"1000원," "500원," "100원" 버튼 중 투입하려는 금액을 선택하세요.

금액 버튼을 클릭하면 투입한 금액이 상단 "투입금액" 영역에 표시됩니다.

3. 음료수 선택하기

자판기에서 원하는 음료수를 선택하세요. 음료수는 표 아래에 나열되어 있습니다.

각 음료수에는 가격이 표시되어 있으며, 버튼을 클릭하여 선택할 수 있습니다.

투입한 금액이 음료수의 가격보다 크면 선택한 음료수가 화면 상단에 나타납니다. 만약 금액이 부족하면 경고 메시지가 표시됩니다.

4. 잔돈 반환하기

음료수를 선택한 후에도 남은 금액이 있으면 "잔돈 반환" 버튼을 클릭하여 잔돈을 반환받을 수 있습니다.

반환된 500원, 100원의 개수가 하단에 표시됩니다.

5. 프로그램 종료하기

자판기 사용을 마치면 웹 브라우저를 닫거나 index.html 파일을 닫아 프로그램을 종료합니다.

> 18. 속성 변경 및 페이지 이동

아래에는 속성을 변경, 추가, 제거하는 메서드의 사용법을 간략하게 설명합니다:

속성 추가하기 (setAttribute):

setAttribute 메서드를 사용하여 요소의 속성 값을 변경할 수 있습니다.

문법: element.setAttribute(attributeName, attributeValue)

element: 속성을 변경할 요소입니다.

attributeName: 변경하려는 속성의 이름을 나타냅니다.

attributeValue: 새로운 속성 값입니다.

```
// id가 "myElement"인 요소의 class 속성을 변경
const element = document.getElementById("myElement");//찾는다.
element.setAttribute("class", "newClass");//변경한다
```

```
//id를 새로운 아이디로 변경
```

```
<div id="myElement">원래의 내용</div>
```

```
<script>
```

```
    // id가 "myElement"인 요소의 id 속성을 변경
```

```
    const element = document.getElementById("myElement");
```

```
    element.setAttribute("id", "newId");
```

```
    console.log("변경된 id: " + element.id);
```

```
</script>
```

```

```

```
<script>
```

```
    // id가 "myImage"인 img 요소의 src 속성을 setAttribute를 사용하여 변경
```

```
    const imageElement = document.getElementById("myImage");
```

```
    imageElement.setAttribute("src", "new-image.jpg");
```

```
</script>
```

```
const element = document.getElementById("example");
```

```
element.setAttribute("style", "color: red;");
```

`element.setAttribute("color", "red");` 이거는 html 속성에 color가 없어서 안된다.

속성 변경하기 (`setAttribute`):

`setAttribute` 메서드를 사용하여 없는 속성을 추가하면 추가되고 이미 존재하면 변경된다. 사용 방법은 속성 추가 방법과 동일하다.

속성 읽어오기 (`getAttribute`):

```
const classValue = element.getAttribute("class");
const idValue = element.getAttribute("id");
const srcValue = imageElement.getAttribute("src");
const hrefValue = linkElement.getAttribute("href");
const styleValue = element.getAttribute("style");
```

속성 제거하기 (`removeAttribute`):

`removeAttribute` 메서드를 사용하여 요소의 속성을 제거할 수 있습니다.

문법: `element.removeAttribute(attributeName)`

`element`: 속성을 제거할 요소입니다.

`attributeName`: 제거하려는 속성의 이름을 나타냅니다.

```
element.removeAttribute("class");
element.removeAttribute("src");
element.removeAttribute("style");
// id가 "myElement"인 요소의 title 속성 제거
const element = document.getElementById("myElement");
element.removeAttribute("title");
```

<!DOCTYPE html> 다음 예제를 확인해 보자.

```
<html lang="en">
```

```
<head>
```

```
  <script>
```

```
    window.onload=function(){
```

```
      var myA=document.getElementById('myA');
```

```
      //var myA=document.getElementById('#myA');
```

```
      //읽어오기
```

```
      console.log(myA.href);
```

```
      console.log(myA.getAttribute("href")); //myA.href
```

```
      //속성 변경하기
```

```
      myA.setAttribute("href","http://www.naver.com");
```

```
      console.log(myA.getAttribute("href")); //myA.href
```

```
      //속성 추가
```

```
      myA.title="naver";
```

```
      //myA.setAttribute("title","naver");
```

```
      console.log(myA.getAttribute("title"));
```

```

        //속성 제거
        myA.removeAttribute("title");
        console.log(myA.getAttribute("title")); //myA.href
        myA.innerHTML="naver";
        //다른페이지로 이동할때 사용
        location.href="http://www.daum.net";
    }
</script>
</head>
<body>
    <a id="myA" href="http://www.daum.net"> daum</a>
</body>
</html>

```

> 19. 시간계산

자바스크립트에서는 Date객체를 이용해서 시간을 관리할 수 있다.

현재 날짜 및 시간 가져오기:

```
var currentDate = new Date();
console.log(currentDate);
```

currentDate에는 현재 날짜와 시간이 포함된 Date 객체가 생성됩니다.

특정 날짜 및 시간으로 객체 생성:

```
var specificDate = new Date("2022-01-01T12:00:00");
console.log(specificDate);
```

특정 날짜와 시간을 나타내는 문자열을 인수로 전달하여 해당 날짜 및 시간에 해당하는 Date 객체를 만들 수 있습니다.

여러 매개변수:

```
var specificDate = new Date(2022, 0, 1);
```

연도, 월 (0부터 시작), 일을 나타내는 여러 개의 매개변수를 전달하여 특정 날짜의 Date 객체를 생성할 수 있습니다.

```
var specificDate = new Date(2022, 0, 1, 12, 0, 0);
```

연도, 월 (0부터 시작), 일, 시, 분, 초를 나타내는 여러 개의 매개변수를 전달하여 특정 날짜의 Date 객체를 생성할 수 있습니다.

타임스탬프 매개변수:

```
var timestampDate = new Date(1646149200000);
```

1970년 1월 1일 자정(UTC)부터의 경과 시간을 밀리초로 Date 객체를 생성할 수 있습니다.

기존 Date 객체 매개변수:

```
var currentDate = new Date();
```

```
var copyDate = new Date(currentDate);
```

기존의 Date 객체를 전달하여 그 객체와 동일한 날짜와 시간을 가지는 새로운 Date 객체를 생성할 수 있습니다.

특정 날짜 및 시간의 각 구성 요소 얻기:

```
var currentDate = new Date();
```

```
var year = currentDate.getFullYear(); // 1970
```

```
var month = currentDate.getMonth(); // 0~11 월은 0부터 시작 실제 월 값에 +1이 필요
```

```
var day = currentDate.getDate(); // 1~
```

```
var hours = currentDate.getHours(); // 0~23
```

```
var minutes = currentDate.getMinutes(); // 0~59
```

```
var seconds = currentDate.getSeconds(); // 0~59
```

```
console.log(year, month + 1, day, hours, minutes, seconds);
```

```
getFullYear(), getMonth(), getDate(), getHours(), getMinutes(), getSeconds()
```

등의 메서드를 사용하여 Date 객체의 각 구성 요소 년월일 시분초를 얻을 수 있습니다.

```
// set메소드로 시간 날짜 설정하기
```

```
var 현재날짜시간 = new Date();
```

```
console.log("현재 날짜:", 현재날짜시간);
```

```
// 다음 달 15일로 날짜 설정
```

```
현재날짜시간.setMonth(현재날짜시간.getMonth() + 1, 15);
```

```
console.log("다음 달 15일로 날짜 설정 후:", 현재날짜시간);
```

```
// 현재 날짜에 3달 더하기
```

```
현재시간.setMonth(현재시간.getMonth() + 3);
```

```
console.log("3달 더한 후 날짜:", 현재시간);
```

```
// 시간을 8시로 설정
```

```
현재날짜시간.setHours(8);
```

```
console.log("시간을 8로 설정 후:", 현재날짜시간);
```

```
// 시간을 18:30:45로 설정
```

```
현재날짜시간.setHours(18, 30, 45);
```

```
console.log("시간을 18:30:45로 설정 후:", 현재날짜시간);
```

```
// 연도를 2023년으로 설정
```

```
현재날짜시간.setFullYear(2023);
```

```
console.log("연도를 2023년으로 설정 후:", 현재날짜시간);
```

```
// 현재 날짜에 3달 더하기
```

```
현재시간.setMonth(현재시간.getMonth() + 3);
```

```
console.log("3달 더한 후 날짜:", 현재시간);
```

```
// 현재 날짜에서 5달 전의 날짜를 계산
```

```
var myDate = new Date();
```

```
console.log("현재 날짜:", myDate);
```

```
myDate.setMonth(myDate.getMonth() - 11);
```

```
console.log("11달 전 날짜:", myDate); //현재 시간이 2050년 이면 2049년이된다.
```

```
"2022-03-07T12:00:00-05:00"
```

ISO 8601 형식은 날짜와 시간을 YYYY-MM-DDTHH:mm:ss의 형태로 표현합니다. 여기서 T는 날짜와 시간을 구분하는 역할을 합니다.

따라서 "2022-03-07T12:00:00-05:00"에서 T 이후의 부분은 시간을 나타내며, -05:00은 시간대를 나타냅니다. 이 경우에는 UTC에서 5시간 미만으로 떨어진 동부 표준시(EST)를 나타냅니다.

getTime()은 JavaScript에서 Date 객체의 메서드 중 하나입니다. 이 메서드는 1970년 1월 1일 00:00:00(UTC)부터 현재까지의 밀리초(milliseconds) 수를 반환합니다.

```
var currentDate = new Date();
var timestamp = currentDate.getTime();
console.log("현재 타임스탬프: " + timestamp);
두 시간차 구하기 밀리세컨드로 만들기
function getTimeStampDifference(date1, date2) {
    // 각 날짜의 타임스탬프 구하기
    var timestamp1 = date1.getTime();
    var timestamp2 = date2.getTime();
    // 두 타임스탬프 간의 차이 구하기 (밀리초 단위)
    var timeDifference = Math.abs(timestamp2 - timestamp1);
    // 차이를 그대로 반환
    return timeDifference;
}
// 예제: 현재 시간과 1시간 전의 차이 계산
var currentDate = new Date();
var oneHourAgo = new Date(currentDate);
oneHourAgo.setHours(currentDate.getHours() - 1);
var timeDiff = getTimeStampDifference(currentDate, oneHourAgo);
console.log("두 시간의 차이(밀리초): " + timeDiff);
```

다음 예제들을 확인해 보자.

--시간계산1

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        var myDate = new Date(2022, 5, 6, 10, 10, 10); // 특정 시간
        var myDate = new Date(); // 현재 시간

        document.write(myDate, "<br>");
        document.write(myDate.getFullYear(), "<br>"); // 1970
        document.write(myDate.getMonth(), "<br>"); // 0~11
        document.write(myDate.getDate(), "<br>"); // 1~
```


*//new Date(0) 에서 0은 기준시간을 의미하고 기준시간에서 1초 지날때마다 1000씩 증가한다. 기준시간보다 1일 이후의 시간을 얻고 자한다면 1000*60*60*24 하면된다.*

```
var myDate=new Date(86400000);//1970년 1월 2일 9시 0분 0초  
document.write(myDate,"<br>");  
//시간 객체에 .getTime() 메소드를 사용하면 기준시간에서 몇 미리세컨드 차이나는지 얻을 수 있다.
```

```
document.write(myDate.getTime(),"<br>")
```

//기준시로 2초지난시간은 다음과 같다.

```
var myDate4=new Date(1970,0,1,9,0,2);  
document.write(myDate4.getTime(),"<br>")  
document.write(new Date(2000),"<br>")  
document.write(24*60*60*1000,"<br>"); //86400000 하루 밀리세컨드  
//결론 시간 객체 2개의 차이 시간을 구하고 싶다면  
//두시간을 getTime메소드를 이용해서 숫자로 변경한후  
//가까운시간에서 먼시간을 빼면 두 시간의 차이를 구할 수 있다.  
//구한 수는 1000이 1초이다.
```

```
var firstTime=new Date(2000,5,6,5,0,0);  
var secondTime=new Date(2000,5,7,5,0,0);  
//.getTime메소드는 시간을 숫자로 리턴해 두시간을 빼서 시간차를 구할 수 있다.  
var diff=secondTime.getTime()-firstTime.getTime();  
document.write(diff,"<br>");
```

//2022-05-06 5:20:15 와 2021-03-02 15:20:15 의 시간차를 구해보자.

```
var myDate1=new Date(2022,4,6,5,20,15);  
document.write(myDate1,"<br>");  
var myDate2=new Date(2021,1,2,15,20,15);  
document.write(myDate2,"<br>");
```

//최근 시간이 큼

```
var diff=myDate1.getTime()-myDate2.getTime();  
var date=diff/(1000*60*60*24);  
var hours=diff/(1000*60*60);  
var minutes=diff/(1000*60);  
var seconds=diff/(1000);
```

```
document.write(parseInt(date),"<br>");  
document.write(parseInt(hours),"<br>");  
document.write(parseInt(minutes),"<br>");  
document.write(parseInt(seconds),"<br>");
```

```
var diff = 12345678; // 예시로 주어진 시간 밀리초  
var seconds = Math.floor(diff / 1000);// 밀리초를 초로 변환  
var minutes = Math.floor(seconds / 60);// 초를 분으로 변환  
seconds %= 60;  
var hours = Math.floor(minutes / 60);// 분을 시간으로 변환
```

```

minutes %= 60;
var days = Math.floor(hours / 24); // 시간을 일로 변환
hours %= 24;

// 결과 출력
console.log(days+"일 "+hours+"시간 "+minutes+"분 "+seconds+"초");
</script>
</head>
<body></body></html>

```

-시간예제 3

startDate시간과 endDate시간 차이를 구하는 예제이다.

startDate시간이 endDate시간 보다 커야한다는 점을 유의하자.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Document</title>
  <script>
    function calculateTimeDifference(startDate, endDate) {
      // 시작 날짜와 종료 날짜에서 연도, 월, 일, 시간, 분, 초 추출
      var startYear = startDate.getFullYear();
      var startMonth = startDate.getMonth() + 1; // 월은 0부터 시작
      var startDay = startDate.getDate();
      var startHours = startDate.getHours();
      var startMinutes = startDate.getMinutes();
      var startSeconds = startDate.getSeconds();

      var endYear = endDate.getFullYear();
      var endMonth = endDate.getMonth() + 1; // 월은 0부터 시작하므로 1을 더함
      var endDay = endDate.getDate();
      var endHours = endDate.getHours();
      var endMinutes = endDate.getMinutes();
      var endSeconds = endDate.getSeconds();
      // 초 차이 계산
      var secondsDifference = endSeconds - startSeconds;
      if (secondsDifference < 0) {
        secondsDifference += 60;
        endMinutes--;
      }
      // 분 차이 계산
      var minutesDifference = endMinutes - startMinutes;
      if (minutesDifference < 0) {
        minutesDifference += 60;
        endHours--;
      }
    }
  </script>

```

```

    }
    // 시간 차이 계산
    var hoursDifference = endHours - startHours;
    if (hoursDifference < 0) {
        hoursDifference += 24;
        endDay--;
    }
    // 일 차이 계산
    var dayDifference = endDay - startDay;
    if (dayDifference < 0) {
        var tempStartDate = new Date(startDate);
        tempStartDate.setMonth(tempStartDate.getMonth() + 1);
        dayDifference = (endDate - tempStartDate)/(1000*60*60* 24);
    }
    // 월 차이 계산
    var monthDifference = endMonth - startMonth;
    if (monthDifference < 0) {
        monthDifference += 12;
        endYear--;
    }
    // 연도 차이 계산
    var yearDifference = endYear - startYear;
    // 결과를 객체로 반환
    var timeDifferenceObj = {
        years: yearDifference,
        months: monthDifference,
        days: dayDifference,
        hours: hoursDifference,
        minutes: minutesDifference,
        seconds: secondsDifference
    };
    return timeDifferenceObj;
}

// 테스트를 위한 예제
var startDate1 = new Date(2022, 4, 6, 5, 20, 15); // 시작 날짜
var endDate1 = new Date(2023, 5, 7, 6, 25, 20); // 종료 날짜
var timeDifference1 = calculateTimeDifference(startDate1, endDate1);

var startDate2 = new Date(2020, 5, 6, 5, 20, 15); // 시작 날짜
var endDate2 = new Date(2021, 6, 7, 6, 25, 20); // 종료 날짜
var timeDifference2 = calculateTimeDifference(startDate2, endDate2);

// 결과 출력
console.log("예제 1 결과:");
console.log("년:", timeDifference1.years);
console.log("월:", timeDifference1.months);
console.log("일:", timeDifference1.days);

```

```

    console.log("시간:", timeDifference1.hours);
    console.log("분:", timeDifference1.minutes);
    console.log("초:", timeDifference1.seconds);

    console.log("예제 2 결과:");
    console.log("년:", timeDifference2.years);
    console.log("월:", timeDifference2.months);
    console.log("일:", timeDifference2.days);
    console.log("시간:", timeDifference2.hours);
    console.log("분:", timeDifference2.minutes);
    console.log("초:", timeDifference2.seconds);
</script>
</head>
<body>
</body>
</html>

```

//1. 사용자에게 구매 시간을 입력받아서 반품가능한지 출력해 보자. 1달이내에 구매해야 반품이 가능하다.

//2. 두시간과 시간당요금을 입력받아 pc방요금을 계산해 보자.

다음 예제는 사용자로 부터 시간데이터를 입력 받기

```

<!DOCTYPE html>
<html>
<head>
  <title>DateTime Input to JavaScript Date Object</title>
</head>
<body>
  <input type="datetime-local" id="datetimeInput">
  <button onclick="convertInputToDateTime()">JavaScript Date</button>
  <script>
    function convertInputToDateTime() {
      // Get the input element by its ID
      var inputElement = document.getElementById("datetimeInput");
      // Get the value (date and time) from the input element
      var dateTimeValue = inputElement.value;
      // Create a JavaScript Date object from the value
      var jsDate = new Date(dateTimeValue);
      // You can now use jsDate for further processing
      console.log("JavaScript Date Object:", jsDate);
    }
  </script>
</body>
</html>

```

-문제풀이

```

<!DOCTYPE html>

```

```

<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    window.onload=function(){
document.getElementById("btn1").addEventListener("click",function(){
  var inputTime=document.getElementById("pDate").value
  //alert(inputTime)
  var inputTime=new Date(inputTime);
  //alert(inputTime);
  //반품 가능한지 여부? 1달 이내이면 반품가능
  //구매한 시간에 1달을 추가한다.
  inputTime.setMonth(inputTime.getMonth()+1);
  //alert(inputTime);
  //1달 추가한 시간이 현재시간보다 크면 반품가능
  if(new Date().getTime() < inputTime.getTime()){
    alert('반품 가능');
  }else{//작으면 반품 불가능
    alert('반품 불가능');
  }
})
document.getElementById("btn2").addEventListener("click",function(){
  var pDate1=new Date(document.getElementById("pDate1").value);
  var pDate2=new Date(document.getElementById("pDate2").value);
  var change =Number(document.getElementById("change").value);
  var diff=pDate2.getTime()-pDate1.getTime();
  var diff=diff/(1000*60*60);
  change=diff*change;
  alert(change);
})
}
</script>
</head>
<body>
  반품가능여부
  구매한시간<input type="date" id="pDate">
  <input type="button" id="btn1" value="button">
  <br>
  PC방요금계산 <br>
  시간당요금<input type="text" id="change" value="1000"/>
  시작시간 <input type="datetime-local" id="pDate1">
  종료시간 <input type="datetime-local" id="pDate2">
  <input type="button" id="btn2" value="button">
</body>
</html>

```

왼쪽처럼 숙박시설 요금계산

휴먼 호텔 숙박 요금 계산 프로그램

체크인: 2024-03-09

체크아웃: 2024-03-10

프로그램을 만들어보자.

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    #fee{
      margin-top: 10px;
    }
  </style>
</head>
<body>
  <header><h1>휴먼 호텔 숙박 요금 계산 프로그램</h1></header>
  <main>
    <label for = "checkInDate">체크인: </label>
    <input type = "date" id = "checkInDate" required><br>
    <label for = "checkOutDate">체크아웃: </label>
    <input type = "date" id = "checkOutDate" required><br>
    <label for = "headCount">인원 수: </label>
    <input type = "number" id = "headCount" required><br>
    <div id = "roomChoice">
      <p>룸 선택</p>
    <input type="radio" name="roomChoice" id="single" value="single" required>싱글
    <input type="radio" name="roomChoice" id="double" value="double" required>더블
    <input type="radio" name="roomChoice" id="twin" value="twin" required>트윈
    <input type="radio" name="roomChoice" id="party" value="party" required>파티
    <input type="radio" name="roomChoice" id="suite" value="suite" required>스위트
    </div>
    <div id = "roomService">
      <p>룸 서비스</p>
    <input type="checkbox" name="roomService" id="breakfast" value="breakfast">조식
    <input type = "checkbox" name = "roomService" id = "lunch" value = "lunch">중식
    <input type = "checkbox" name = "roomService" id = "dinner" value = "dinner">저녁
```

```

<input type = "checkbox" name = "roomService" id = "snack" value = "snack">과자
<input type = "checkbox" name = "roomService" id = "beverage" value = "beverage">술
<input type="checkbox" name="roomService" id="morningCall" value="morningCall">모닝콜
    </div>
</main>
<div id = "fee">
    <button onclick="no3()">숙박 요금 계산</button>
    <input type = "text" id = "displayMoney" value = "0원" readonly>
</div>
</body>
<script>
    function no3(){
        //각각의 값 입력 받기
        const checkInDateInput = document.getElementById("checkInDate");
        const checkOutDateInput = document.getElementById("checkOutDate");
        const headCountInput = document.getElementById("headCount");
        const roomChoiceInput = document.getElementsByName("roomChoice");
        const roomServiceInputs =
document.querySelectorAll('input[name="roomService"]:checked');
        //룸 가격 정보
        const roomPrices = {
            single: 100000,           double: 150000,
            twin: 120000,           party: 200000,           suite: 250000
        };
        //룸 선택 값 가져오기
        const roomType =
document.querySelector('input[name="roomChoice"]:checked').value;
        const roomPrice = roomPrices[roomType];
        //체크인, 체크아웃 값 date객체로 변환
        const checkInTime = new Date(checkInDateInput.value);
        const checkOutTime = new Date(checkOutDateInput.value);
        //사용 날짜 계산
        const timeDiff = checkOutTime - checkInTime;
        //기본 숙박 요금 계산
        const baseFee = (timeDiff / (1000 * 60 * 60 * 24))*roomPrice;
        //룸 서비스 요금 정보
        const servicePrices = {
            breakfast: 20000,           lunch:40000,
            dinner:60000,           snack:30000,
            beverage:50000,           morningCall:0
        }
        //룸 서비스 가격 계산
        let totalRoomServicePrice = 0;
        for(let i = 0; i < roomServiceInputs.length; i++){
            const serviceName = roomServiceInputs[i].value;
            const servicePrice = servicePrices[serviceName];
            totalRoomServicePrice += servicePrice;
        }
    }

```



```

        //올바른 값 입력 확인
        if(!checkInDateInput.value || !checkOutDateInput.value ||
!headCountInput.value || !roomType || !roomServiceInputs.length){
            alert("올바른 값을 입력하세요");
            return;
        }
        //총 가격 출력
        const totalFee= baseFee+totalRoomServicePrice;
        document.getElementById('displayMoney').value=totalFee/10000+ '만원';
    }
</script>
</html>

```

> 20. 배열 예제 1

이 예제는 0과 1로 이루어진 이차원 배열을 사용하여 HTML 테이블을 생성하고, 배열의 값에 따라 테이블 셀의 배경색을 조작하는 것을 보여주는 예제입니다.

tableArray: 0과 1로 구성된 이차원 배열로, 1은 셀의 배경색을 빨간색으로, 0은 셀의 배경색을 흰색으로 표현합니다.

madeTable(y, x): 이 함수는 y 행과 x 열의 HTML 테이블을 문자열 형태로 생성합니다.

테이블은 각 셀에는 id를 부여하여 추후에 해당 셀을 식별할 수 있도록 합니다.

drawArray(arr): 이 함수는 주어진 이차원 배열 arr의 값을 검사하여, 배열 내의 1에 해당하는 위치의 셀의 배경색을 빨간색으로 변경합니다.

window.onload: 페이지가 로드될 때 실행되며, 먼저 madeTable 함수를 사용하여 HTML 테이블을 생성하고, 그 다음 drawArray 함수를 사용하여 tableArray에 정의된 패턴에 따라 테이블의 셀 배경색을 설정합니다.

document.getElementById("y"+i+"x"+j).setAttribute("bgcolor","red"); 특정 id의 td를 찾아 배경속성을 빨간색으로 변경한다.

다음 코드를 확인해 보자.

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script>
        var tableArray=[
            [1,0,1,1,1,1,1,1,0,1],
            [1,0,1,0,1,1,1,0,1,1],
            [1,0,1,1,0,1,0,1,1,1],
            [1,0,1,1,1,0,1,1,1,1],
            [1,0,1,1,0,1,0,1,1,1],
            [1,0,1,0,1,1,1,0,1,1],

```

```

        [1,0,0,1,1,1,1,1,0,1],
        [1,0,1,0,1,1,1,1,1,0],
        [1,0,1,1,0,1,1,1,0,1],
        [1,0,1,1,1,0,1,0,1,1],
        [1,0,1,1,1,1,0,1,1,1]
    ];
    //y,x만큼 html table td를 만든다.
    function madeTable(y,x){
        var str="";
        str+="




```

> 21. 배열예제 2



다음을 만들어 보자. HTML 테이블의 셀을 동적으로 교체하여 전광판 효과를 만드는 예제입니다. 페이지가 로드되면 초기 상태의 왼쪽과 같은 테이블이 나타나며, 1초마다 move 함수가 호출되어 색상이 오른쪽으로 한 칸씩 이동한다.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    function move(){
      var temp=document.getElementById("x2").getAttribute("bgcolor");
      console.log(temp);
      for(var i=2;i>0;i--){
        document.getElementById("x"+i).setAttribute("bgcolor",
          document.getElementById("x"+(i-1)).getAttribute("bgcolor"));

        // document.getElementById("x1").setAttribute("bgcolor",
        // document.getElementById("x0").getAttribute("bgcolor"));
      }
      document.getElementById("x0").setAttribute("bgcolor",temp);
    }
    window.onload=function(){
      // var temp=document.getElementById("x2").getAttribute("bgcoLor");
      // console.log(temp);
      // document.getElementById("x2").setAttribute("bgcoLor",
      // document.getElementById("x1").getAttribute("bgcoLor"));

      // document.getElementById("x1").setAttribute("bgcoLor",
      // document.getElementById("x0").getAttribute("bgcoLor"));
```

```

        // document.getElementById("x0").setAttribute("bgcolor",temp);
        var intervalId=setInterval(move,1000);
    }
    //전광판
</script>
</head>
<body>
    <table border="1" width="300" height="100">
        <tr>
            <td id="x0" bgcolor="red"></td><td id="x1"
bgcolor="blue"></td><td bgcolor="green" id="x2"></td>
        </tr>
    </table>
</body>
</html>

```

HTML 프로그램 사용자 매뉴얼

전광판은 가로 및 세로 크기에 따라 격자 모양으로 나타납니다.

다음 방향으로 테이블을 이동하려면 해당 방향 버튼

"위로 움직이기" 버튼: 테이블을 위로 이동합니다.

"아래로 움직이기" 버튼: 테이블을 아래로 이동합니다.

"왼쪽으로 움직이기" 버튼: 테이블을 왼쪽으로 이동합니다.

"오른쪽으로 움직이기" 버튼: 테이블을 오른쪽으로 이동합니다.

"정지" 버튼을 클릭하여 테이블 이동을 멈출 수 있습니다.

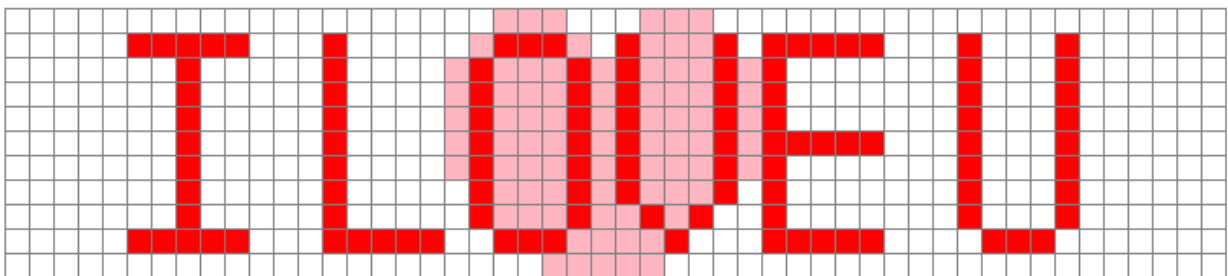
이동 속도 조절하기

"속도 증가" 버튼과 "속도 감소" 버튼을 사용하여 테이블 이동 속도를 조절할 수 있습니다.

"속도 증가" 버튼을 클릭하면 테이블 이동 속도가 빨라집니다. (최대 200ms)

"속도 감소" 버튼을 클릭하면 테이블 이동 속도가 느려집니다. (최소 10ms)

만드는 방법 배열에 아래 이미지 처럼 표현할 수 있도록 0,1,2를 넣는다. 타이머를 이용해서 선택한 방향으로 글씨가 움직이게 한다. interval를 이용해서 타이머 속도를 변경하자.



위로움직이기 아래로움직이기

왼쪽으로움직이기 오른쪽으로움직이기

정지

속도증가 속도감소

다음 코드를 참고하자.

```
<!-- JavaScript 코드 일부분 -->
// 초기 2차원 배열
var tableData = [
    // ...
];
// 초기 상태의 테이블을 그리는 함수
function madeTable(y, x, arr) {
    // ...
}
// tableData를 매개변수로 받아 들어 있는 데이터 숫자에 맞춰서 색칠하는 함수
function drawTable( arr) {
    // ...
}
// 왼쪽으로 이동하는 함수
function moveLeft() {
}
// 오른쪽으로 이동하는 함수
function moveRight() {
}
// 위로 이동하는 함수
function moveUp() {
}
// 아래로 이동하는 함수
function moveDown() {
}
// 이동을 멈추는 함수
function stopMovement() {
    // 모든 이동 방향의 interval을 정지하는 로직을 추가하세요
}
// 초기 상태로 리셋하는 함수
function resetTable() {
// tableMove 배열을 초기 상태로 되돌리고 테이블을 다시 그리는 코드를 작성하세요
}

// 버튼 클릭 이벤트에 함수 연결
document.getElementById("left").onclick = moveLeft;
document.getElementById("right").onclick = moveRight;
```

```
document.getElementById("up").onclick = moveUp;
document.getElementById("down").onclick = moveDown;
document.getElementById("stop").onclick = stopMovement;
document.getElementById("reset").onclick = resetTable;
```

> 22. DOM를 이용한 CSS 변경

다음과 예제를 확인해서 자바스크립트에 속성 변경하는 방법을 확인해 보자. 버튼을 누르면 상위 화면이 아래 화면으로 변경된다. 버튼을 여러분 눌러보자.

DOM (Document Object Model)을 사용하여 웹 페이지의 엘리먼트를 찾고 해당 엘리먼트 객체의 style속성을 이용해서 css를 변경할 수 있다.

```
// DOM을 사용하여 엘리먼트 찾기
var paragraphElement = document.getElementById('exampleParagraph');

// 엘리먼트의 style 속성을 변경하여 CSS 적용
paragraphElement.style.color = 'blue';
paragraphElement.style.fontSize = '20px';
```

다음 예제는 버튼을 클릭하면 상위 이미지가 하위 이미지처럼 css가 변경 된다. 예제를 확인해 보자.



<!DOCTYPE html>

```

<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    #div1{
      border: 1px solid blue;
    }
  </style>
  <script>
    // 변경하기
    //html태그 style속성을 이용해서 javascript로 css를 변경할 수 있다.
    window.onload=function(){
      var myButton=document.getElementById("b1");
      myButton.addEventListener("click",function(){
        var myDiv=document.getElementById("div1");
        myDiv.style.backgroundColor='yellow';
        myDiv.style.width="400px";
        myDiv.style.height='100px';
        //클릭할때마다 높이를 50px 증가하는 작업을 하기 위해서 px와 같은 단위
        삭제하고 추가하는 작업을 하고 있다.
        console.log(myDiv.style.height);
        console.log(parseInt(myDiv.style.height));
        myDiv.style.height=parseInt(myDiv.style.height)+50+"px";
        //hello1를 빨간색으로 바꿔보자.
        var mySpan=document.querySelectorAll("#div1 span");
        console.log(mySpan);
        mySpan[0].style.color='red';
        //document.getElementsByTagName("span");//document.에서 span를 2개 찾음
        //myDiv.getElementsByTagName("span");//myDiv에서 span를 찾음 1개찾음
        // style="height:100px" 이부분을 삭제하면 에러가 발생하는 이유는
      })
    }
  </script>
</head>
<body>
  <div id="div1" style="height:100px">
    <span>hello1</span> world
  </div>
  <button id="b1"><span>change style</span></button>
</body>
</html>

```

다음 경마 프로그램을 구현해 보자.

이 코드는 HTML, CSS 및 JavaScript를 사용하여 간단한 경마 게임을 만드는 예제입니다. 사용자가 "start" 버튼을 클릭하면 말들이 경주를 시작하고, 각각의 말들은 랜덤한 속도로 오른쪽으로 이동합니다. 말들 중 하나가 화면 오른쪽 끝에 도달하면 경주가 종료되며, 각 말이

몇 번째로 도착했는지 알려주는 경주 순위가 표시됩니다.

힌트: 타이머를 이용해서 DIV안에 있는 말 이미지의 `marginLeft` 값을 변경하다 먼저 값이 800에 도달한 말 순으로 순위를 정한다.

`document.getElementById("h1").style.marginLeft`



코드의 주요 부분을 설명하겠습니다:

HTML 부분:

"start" 버튼: 사용자가 게임을 시작하는 데 사용됩니다.

`<div id="line">`: 경주 말들이 이동하는 경주 트랙을 나타냅니다.

각 말은 `` 요소로 표시되고 각각 고유한 ID(h1, h2, h3, h4)를 가집니다.

CSS 부분:

#line 스타일: 트랙의 가로 선을 그리기 위해 `border-right` 속성을 사용합니다.

JavaScript 부분:

start() 함수: "start" 버튼을 클릭할 때 호출되며, 경주가 시작됩니다. setInterval 함수를 사용하여 gaming() 함수가 일정한 시간 간격으로 호출됩니다.

gaming() 함수: 경주 말들을 이동시키는 주요 로직이 들어 있습니다.

Math.random() 함수를 사용하여 말들의 랜덤한 이동을 시뮬레이션합니다.

각 말의 위치가 화면 오른쪽 끝(800px)에 도달하면, 해당 말은 경주가 종료된 것으로 간주합니다. 그리고 rank 배열에 도착한 순서(1, 2, 3, 4)를 기록합니다.

모든 말이 도착하면 경주가 종료되고, 경주 결과가 alert로 표시됩니다.

경주가 종료되면 "start" 버튼이 다시 표시되어 게임을 재시작할 수 있도록 합니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    #line{
      width:800px;
      border-right:solid 2px black;
    }
  </style>
  <script>
```



```

//가진돈 건돈
//var totalMoney=10000;
// var isEnd=[false,false,false,false];//false 안도착 true 도착
// var rank=[0,0,0,0];           //도착순위
// var rankCount=1;              //순위설정
var rank=[];
function start(){
    document.getElementById("start").style.display='none';
    // isEnd=[false,false,false,false];//false 안도착 true 도착
    // rank=[0,0,0,0];           //도착순위
    // rankCount=1;              //순위설정
    rank=[];
    document.getElementById("h1").style.marginLeft='0px';
    document.getElementById("h2").style.marginLeft='0px';
    document.getElementById("h3").style.marginLeft='0px';
    document.getElementById("h4").style.marginLeft='0px';
    timerId=setInterval(gaming,100);
}
function gaming(){
    document.getElementById("h1").style.marginLeft=parseInt(
document.getElementById("h1").style.marginLeft)+parseInt(Math.random()*50)+'px';
    document.getElementById("h2").style.marginLeft=parseInt(
document.getElementById("h2").style.marginLeft)+parseInt(Math.random()*50)+'px';
    document.getElementById("h3").style.marginLeft=parseInt(
document.getElementById("h3").style.marginLeft)+parseInt(Math.random()*50)+'px';
    document.getElementById("h4").style.marginLeft=parseInt(
document.getElementById("h4").style.marginLeft)+parseInt(Math.random()*50)+'px';

    if(parseInt(document.getElementById("h1").style.marginLeft)>800){
        document.getElementById("h1").style.marginLeft='810px';
        if(!rank.includes(1)){//1번말이 이미 도착했는 확인해서
            rank.push(1);//1번말이 없으면 1번말을 추가한다.
        }
        // if(!isEnd[0]){
        //     isEnd[0]=true; rank[0]=rankCount; rankCount++;
        // }
    }
    if(parseInt(document.getElementById("h2").style.marginLeft)>800){
        document.getElementById("h2").style.marginLeft='810px';
        if(!rank.includes(2)){
            rank.push(2);

```

```

    }
    // if(!isEnd[1]){
    //     isEnd[1]=true; rank[1]=rankCount; rankCount++;
    // }
}
if(parseInt(document.getElementById("h3").style.marginLeft)>800){
    document.getElementById("h3").style.marginLeft='810px';
    if(!rank.includes(3)){
        rank.push(3);
    }
    // if(!isEnd[2]){
    //     isEnd[2]=true; rank[2]=rankCount; rankCount++;
    // }
}
if(parseInt(document.getElementById("h4").style.marginLeft)>800){
    document.getElementById("h4").style.marginLeft='810px';
    if(!rank.includes(4)){
        rank.push(4);
    }
    // if(!isEnd[3]){
    //     isEnd[3]=true; rank[3]=rankCount; rankCount++;
    // }
}
if(parseInt(document.getElementById("h1").style.marginLeft)>800&&
parseInt(document.getElementById("h2").style.marginLeft)>800&&
parseInt(document.getElementById("h3").style.marginLeft)>800&&
parseInt(document.getElementById("h4").style.marginLeft)>800){
    clearInterval(timerId);
    alert(rank);
    document.getElementById("start").style.display='inline';
}
}
</script>
</head>
<body>
<button onclick="start()" id=start>start</button><br>
<div id=line>
    <br>
    <br>
    <br>
    <br>

```

```
</div>
</body>
</html>
```

> 23. javascript ClassList 조작

classList는 HTML 요소의 클래스를 조작하는 JavaScript 객체입니다. classList 객체를 사용하면 JavaScript를 통해 클래스를 동적으로 추가, 제거, 토글할 수 있다.

classList 객체는 다음과 같은 주요 메서드와 속성을 제공합니다:

add(class1, class2, ...): 클래스 목록에 하나 이상의 클래스를 추가합니다. 만약 클래스가 이미 요소에 존재한다면 중복되지 않습니다.

remove(class1, class2, ...): 클래스 목록에서 하나 이상의 클래스를 제거합니다. 클래스가 요소에 없어도 에러가 발생하지 않습니다.

toggle(class): 클래스를 토글합니다. 토글의 의미는 클래스가 이미 요소에 있는 경우 제거하고, 없는 경우 추가합니다.

contains(class): 클래스가 요소에 있는지 여부를 확인합니다. 클래스가 요소에 있으면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

length: 클래스 목록에 포함된 클래스의 수를 반환합니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>DOM 예제</title>
  <style>
    .highlight {
      color: red;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <p id="exampleParagraph">이것은 예제 문단입니다.</p>
  <script>
    // DOM을 사용하여 엘리먼트 찾기
    var paragraphElement = document.getElementById('exampleParagraph');
    // 엘리먼트의 style 속성을 변경하여 CSS 적용
    paragraphElement.style.color = 'blue';
    paragraphElement.style.fontSize = '20px';
    // 클래스를 추가하여 CSS 클래스 적용
```

```

        paragraphElement.classList.add('highlight');
    </script>
</body>
</html>

```

다음 예제는 classList를 이용해서 동적으로 원하는 클래스를 추가하는 예제이다.

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        .myStyle1{
            width:100%;
            background-color: red;
            color:white;
            font-size: 25px;
            display: block;
        }
        .myStyle2{
            width:100%;
            background-color: blue;
            color:white;
            font-size: 25px;
            display: block;
        }
        .myStyle3{
            width:100%;
            background-color: green;
            color:white;
            font-size: 25px;
            display: block;
        }
        div{
            display: none;
        }
    </style>
    <script>
        window.onload=function(){
            document.getElementById("b1")
                .addEventListener("click",function(){
                    document.getElementById("d1").classList.toggle("myStyle1");
                })
            document.getElementById("b2")
                .addEventListener("click",function(){
                    document.getElementById("d1").classList.add("myStyle2");
                    document.getElementById("d1").classList.add("myStyle3");
                })
        }
    </script>

```

```

        document.getElementById("b3")
        .addEventListener("click",function(){
            document.getElementById("d1").classList.remove("myStyle3");
        })
    }
    // <div id="d1" style="display: none">
    // inline style시트가 우선순위가 높아서 정상동작하지 않는다.
</script>
</head>
<body>
    <button id="b1">toggle</button>
    <button id="b2" class="active">add</button>
    <button id="b3">delete</button>
    <div id="d1">
        DIV 태그
    </div>
</body>
</html>

```

> 24. button를 사용한 active표현하기



여러개의 버튼 중에 하나의 버튼만 선택되게 하는 예제를 만들어 보자.

이 HTML 및 JavaScript 코드는 버튼을 클릭할 때 버튼의 활성 상태를 토글하는 간단한 예제입니다. 버튼을 클릭하면 모든 버튼에서 "active" 클래스가 제거되고 클릭된 버튼에 "active" 클래스가 추가됩니다. 이를 통해 클릭된 버튼을 강조 표시할 수 있습니다. active 클래스가 추가된 버튼은 검정색이고 아닌 버튼은 하늘색 입니다.

HTML 구조:

페이지에는 여러 개의 버튼이 있습니다. 선택된 버튼은 "active" 클래스를 가지고 있습니다.

CSS 스타일:

스타일 시트에서 버튼의 스타일을 정의합니다. 초기 상태에서 "active" 클래스를 가진 버튼의 배경색은 검정색이고 텍스트 색상은 초록색입니다. 마우스를 버튼 위로 올리면 버튼의 배경색이 검정색이 되고 텍스트 색상은 초록색이 됩니다.

JavaScript:

window.onload 이벤트 핸들러는 페이지가 로드되면 실행됩니다. 이 핸들러는 모든 버튼 요소를 가져와서 각 버튼에 대해 클릭 이벤트 리스너를 추가합니다.

클릭 이벤트 리스너는 클릭된 버튼을 강조 표시하기 위해 "active" 클래스를 해당 버튼에 추가하고, 모든 다른 버튼에서 "active" 클래스를 제거합니다.

this.classList.add("active")는 현재 클릭된 버튼에 "active" 클래스를 추가하는 부분입니다.

buttons[j].classList.remove("active")는 모든 버튼에서 "active" 클래스를 제거하는 부분입니다.

결과:

페이지를 열면 초기 상태에서 "2" 버튼이 "active" 클래스를 가지고 있으므로 검정색 배경과 초록색 텍스트를 가집니다.

다른 버튼을 클릭하면 해당 버튼이 "active" 클래스가 추가되어 검정바탕 버튼이 되고 나머지 버튼은 "active" 클래스가 제거되어 일반 스타일로 돌아갑니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    button{/*button관련 active*/
      border: none;
```

```

padding:10px 20px;
background-color: aqua;
cursor: pointer;
}
.active, button:hover{
background-color: black;
color:aqua;
}
</style>
<script>
window.onload=function(){
var buttons=document.getElementsByTagName("button");
for(var i=0;i<buttons.length;i++){//모든 버튼에 이벤트 추가
buttons[i].addEventListener("click",function(){
//모든 버튼에 active속성을 삭제한다.
for(var j=0;j<buttons.length;j++){
buttons[j].classList.remove("active");
}
this.classList.add("active");
});
}
}
</script>
</head>
<body>
<button>1</button><button
class="active">2</button><button>3</button><button>4</button><button>6</button>
<button>7</button>
</body>
</html>

```

> 26. 변수의 scope

JavaScript에서 변수를 선언하는 방법은 var, let, const 세 가지이고 사용방법은 동일 하다.

3개의 차이점은 변수의 생명주기가 var는 함수 단위 스코프이고, let과 const는 블록단위 스코프이다. const로 선언한 변수는 상수여서 변경이 불가능하다.

var 변수 선언:

var를 사용하여 변수를 선언할 때 변수는 함수 범위(scope)를 가집니다.

함수 단위 스코프란? 함수 내에서 선언된 var 변수는 함수가 종료될 때까지 존재하며, 함수 외부에서는 접근할 수 없습니다.

```
function example() {  
    var x = 10;  
    if (true) {  
        var y = 20;  
    }  
    console.log(x); // 10  
    console.log(y); // 20 자바는 기본이 블록 범위에서 다음이 출력되지 않는다.  
}
```

let 변수 선언:

let을 사용하여 변수를 선언할 때 변수는 블록 범위를 가집니다.

블록단위 스코프란? 블록 내에서 선언된 let 변수는 블록을 빠져나가면 메모리에서 삭제되며, 블록 외부에서는 접근할 수 없습니다.

```
function example() {  
    let x = 10;  
    if (true) {  
        let y = 20;  
    }  
    console.log(x); // 10  
    console.log(y); // ReferenceError: y is not defined 자바와 같다.  
}
```

const 변수 선언:

const를 사용하여 변수를 선언할 때 변수 역시 블록 범위를 가집니다.

const 변수는 상수로 간주되어 재할당이 불가능하며, 선언과 동시에 초기화해야 합니다.

```
function example() {  
    const x = 10;  
    if (true) {  
        const y = 20;  
    }  
    //x=20; 에러가 발생한다.  
    console.log(x); // 10  
    console.log(y); // ReferenceError: y is not defined  
}
```


자바스크립트 전역변수:

자바스크립트에서 전역변수를 선언하는 방법은 변수명만으로 변수 선언을 하면 된다.

자바스크립트 전역변수는 window객체에서 관리하고 있어서 전역변수로 선언된 변수는 window.변수명으로 접근할 수 있다.

변수명 없이 변수를 선언하면 앞에 window.이 생략된 것처럼 인식되어 전역변수로 사용할 수 있다. 아래 예제에서 a,b,c가 전역변수 처럼 사용되는 것을 확인 할 수 있다.

```
a = 10;
function example() {
    b=20;
    if(true){
        c=30;
    }
    console.log(a);    console.log(b);    console.log(c);
}
console.log(a);
//example함수가 실행되지 않아서 아래 2개는 생성되어 있지 않다.
//console.log(b);    //console.log(c); // error 발생
//함수는 호출되어야 실행되므로 example()호출후 전역변수가 생성되어 접근 가능함
example()
console.log(window.a); console.log(window.b); console.log(window.c);//접근가능
```

정리

변수선언 방법

var, let, const, 변수명만으로 변수선언

var 함수스코프 형태의 변수 선언

let 블록스코프 형태의 변수 선언

const 블록스코프 형태의 상수 선언

변수명만으로선언 전역변수선언

> 27. 슬라이더 만들기



아래 코드는 이미지 슬라이더를 만드는 HTML과 JavaScript 코드입니다.

일반적으로 전문 프로그래머들이 일반 사용자에게
본인이 만든 자바스크립트를 제공할 때
자바스크립트의 내용을 몰라도 html만으로
자바스크립트 파일을 사용할 수 있도록 제공
한다. 다음에 제공하는 슬라이더 예제도
자바스크립트 내용을 몰라도 html 문서만 보고
자바스크립트 슬라이드 기능을 사용할 수 있다.
보통 자바 스크립트를 모르는 사람에게
자바스크립트를 제공할 때 이런 형태로 제공한다

html:

슬라이더에 필요한 자바스크립트 외부 연결 파일은 `mySlider.js`이다.

Div id가 mySlider에 이미지 슬라이더를 만든다.

div 속성 data-imglist에 슬라이드 이미지를 생성한다.

이전 다음 이미지로 이동하도록 설계되어 있는 a태그가 추가되어 있다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <!-- 외부 자바스크립트 파일 참조 -->
    <script src="js/mySlider.js"></script>
</head>
<body>
    <!-- 이미지 슬라이더를 담은 컨테이너 data-imglist속성으로 슬라이더에 쓸 이미지를 설정한다.-->
    <div id="mySlider" data-imglist=["'img/1.jpg','img/2.jpg','img/3.jpg']">
        <!-- 페이지 상단에 표시될 제목 -->
        <h1>내가만든 이미지 슬라이더</h1>
        <!-- 현재 이미지를 표시할 이미지 요소 -->
        <br>
        <!-- 이미지 이전으로 이동하는 링크(버튼) -->
        <a href="#" id="prev">이전</a> &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
        <!-- 이미지 다음으로 이동하는 링크(버튼) -->
```

```

        <a href="#" id="next">다음</a>
    </div>
</body>
</html>

```

JavaScript (myslider.js):

window.onload 이벤트를 사용하여 HTML 문서가 로드될 때 코드를 실행합니다.
html에 정의되어 있는 태그들을 찾아서 자바스크립트 슬라이더 기능을 추가 하였다.

```

window.onload=function(){
    //var img=['img/1.jpg','img/2.jpg','img/3.jpg'];
    var sliderImgs=document
    .getElementById("mySlider")
    .getAttribute("data-imglist");
    //슬라이더에서 사용할 이미지 배열 제작
    sliderImgs=eval(sliderImgs);
    console.log(sliderImgs);
    console.log(sliderImgs[0]);
    //슬라이더를 실행할 이미지 얻어오기
    var mySlider=document.getElementById("holder");
    var index=0;

    document.getElementById("prev").addEventListener("click",
    function(){

        if(index==0){
            index=sliderImgs.length;
        }
        index=index-1;
        mySlider.src=sliderImgs[index];
        //mySlider.src=sliderImgs[--index];
        // alert(mySlider.src+ ":"+index) src와 index정보확인
        //기본 이벤트가 있을때 무시하고 싶으면 리턴값을 false로 한다.
        return false;
    })
    document.getElementById("next").addEventListener("click",function(){
        index=index+1;
        index=index%sliderImgs.length;
        //index=++index%sliderImgs.length;
        mySlider.src=sliderImgs[index];
    })
}

```

마지막으로, "https://www.w3schools.com/howto/howto_js_slideshow.asp"에서 제공하는
개선된 슬라이드 쇼의 다음 예제를 참고해보자.

```

<!DOCTYPE html>
<html>

```

```

<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
* {box-sizing: border-box}
body {font-family: Verdana, sans-serif; margin:0}
.mySlides {display: none}
img {vertical-align: middle;}
/* Slideshow container */
.slideshow-container {
  max-width: 1000px;
  position: relative;
  margin: auto;
}
/* Next & previous buttons */
.prev, .next {
  cursor: pointer;
  position: absolute;
  top: 50%;
  width: auto;
  padding: 16px;
  margin-top: -22px;
  color: white;
  font-weight: bold;
  font-size: 18px;
  transition: 0.6s ease;
  border-radius: 0 3px 3px 0;
  user-select: none;
}
/* Position the "next button" to the right */
.next {
  right: 0;
  border-radius: 3px 0 0 3px;
}
/* On hover, add a black background color with a little bit see-through */
.prev:hover, .next:hover {
  background-color: rgba(0,0,0,0.8);
}
/* Caption text */
.text {
  color: #f2f2f2;
  font-size: 15px;
  padding: 8px 12px;
  position: absolute;
  bottom: 8px;
  width: 100%;
  text-align: center;
}

```

```

/* Number text (1/3 etc) */
.numbertext {
  color: #f2f2f2;
  font-size: 12px;
  padding: 8px 12px;
  position: absolute;
  top: 0;
}
/* The dots/bullets/indicators */
.dot {
  cursor: pointer;
  height: 15px;
  width: 15px;
  margin: 0 2px;
  background-color: #bbb;
  border-radius: 50%;
  display: inline-block;
  transition: background-color 0.6s ease;
}
.active, .dot:hover {
  background-color: #717171;
}
/* Fading animation */
.fade {
  animation-name: fade; /* @keyframes fade 에 값을 가져와 애니메이션을 만든다. */
  animation-duration: 1.5s;
}
@keyframes fade {
  from {opacity: .4}
  to {opacity: 1}
}
/* On smaller screens, decrease text size */
@media only screen and (max-width: 300px) {
  .prev, .next, .text {font-size: 11px}
}
</style>
</head>
<body>
<div class="slideshow-container">
<div class="mySlides fade">
  <div class="numbertext">1 / 3</div>
  
  <div class="text">Caption Text</div>
</div>
<div class="mySlides fade">
  <div class="numbertext">2 / 3</div>

```

```

    
    <div class="text">Caption Two</div>
</div>
<div class="mySlides fade">
    <div class="numbertext">3 / 3</div>
    
    <div class="text">Caption Three</div>
</div>
<a class="prev" onclick="plusSlides(-1)"><</a>
<a class="next" onclick="plusSlides(1)">>>/a>
</div>
<br>
<div style="text-align:center">
    <span class="dot" onclick="currentSlide(1)"></span>
    <span class="dot" onclick="currentSlide(2)"></span>
    <span class="dot" onclick="currentSlide(3)"></span>
</div>
<script>
let slideIndex = 1;
showSlides(slideIndex);
function plusSlides(n) {
    showSlides(slideIndex += n);
}
function currentSlide(n) {
    showSlides(slideIndex = n);
}
function showSlides(n) {
    let i;
    let slides = document.getElementsByClassName("mySlides");
    let dots = document.getElementsByClassName("dot");
    if (n > slides.length) {slideIndex = 1}
    if (n < 1) {slideIndex = slides.length}
    for (i = 0; i < slides.length; i++) {
        slides[i].style.display = "none";
    }
    for (i = 0; i < dots.length; i++) {
        dots[i].className = dots[i].className.replace(" active", "");
    }
    slides[slideIndex-1].style.display = "block";
    dots[slideIndex-1].className += " active";
}
</script>
</body>
</html>

```

> 28. DOM를 이용한 엘리먼트 생성

사용자가 DOM를 만들어 기존 html에 추가 할 수 있다.

`window.onload = function() { ... }`: 이 부분은 페이지가 로드된 다음 정의된 JavaScript 코드를 실행 한다. 여기서의 코드는 화면에 버튼을 생성하는 코드이다.

`var jbBtn = document.createElement('button');`: `document.createElement`를 사용하여 `<button>` 엘리먼트를 동적으로 생성합니다. 이 버튼은 변수 `jbBtn`에 할당됩니다.

`var jbBtnText = document.createTextNode('Click');`: `document.createTextNode`를 사용하여 텍스트 노드를 생성하고 그 내용을 "Click"으로 설정합니다. 이 텍스트 노드는 변수 `jbBtnText`에 할당됩니다.

`jbBtn.innerHTML = "test";`: `jbBtn` 버튼의 `innerHTML` 속성을 설정하여 버튼 내부에 HTML을 추가합니다. 여기서는 "test" 텍스트를 `` 태그로 감싸 볼드체로 표시하고 있습니다.

`jbBtn.appendChild(jbBtnText);`: 앞서 생성한 텍스트 노드를 버튼 내에 추가합니다. 이것은 "Click" 텍스트를 버튼 내부에 표시하는 역할을 합니다.

`document.body.appendChild(jbBtn);`: `jbBtn` 버튼을 `<body>` 엘리먼트의 자식으로 추가하여 페이지에 표시 됩니다.

결과적으로, 페이지가 로드되면 "Click"이라는 버튼이 나타나며, 버튼 내의 "test" 텍스트는 볼드체로 표시됩니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    window.onload=function(){
      var jbBtn = document.createElement( 'button' );
      var jbBtnText = document.createTextNode( 'Click' );
      jbBtn.innerHTML="<b>test</b>";
      jbBtn.appendChild( jbBtnText );
      document.body.appendChild(jbBtn);
      //innerHTML로 문자열로 된 html태그를 appendChild를 사용하여 바디의 마지막 자식으로
      추가 하였다.
    }
  </script>
</head>
<body> </body>
</html>
```

> 32. DOM 조작

```
<div>
  <p>과일리스트</p>
  <ul>
    <li>사과</li>
    <li>오렌지</li>
    <li>바나나</li>
  </ul>
</div>
```

다음 HTML 문서에는 JavaScript를 사용하여 문서 객체 모델(DOM)을 조작하는 스크립트가 포함되어 있다. 아래 코드와 비교하면서 확인해 보자.

이런 형태의 dom조작을 javascript자체로 많이 사용하지 않으므로 이런것이 있구나 정도 생각 하면 될것 같다. dom 조작은 나중에 배울 jquery를 더 많이 사용한다.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script>
    /*
    DOM이란? Document Object Model 문서를 객체로 만들어 놓은 것
    DOM를 이용해서 노드에 접근하여 변경할 수 있고, 다른 노드를 찾을 수 있다.
    */
    window.onload = function() {
      // <body> 요소의 모든 자식을 변수 body에 저장하고 콘솔에 출력
      var body = document.body.children;
      console.log(body);
      // 첫 번째 <ul> 요소를 변수 myUl에 저장하고 콘솔에 출력
      var myUl = document.getElementsByTagName('ul')[0];
      console.log(myUl);
      //<ul>요소의 모든 자식(<li>요소들)을 변수 myLi에 저장하고 콘솔에 출력
      var myLi = myUl.children;
      console.log(myLi);

      // 모든 <li> 요소의 innerHTML을 순회하며 콘솔에 출력
      for (var i = 0; i < myLi.length; i++) {
        console.log(myLi[i].innerHTML);
      }

      // <ul> 요소의 첫 번째 자식 노드의 텍스트를 콘솔에 출력
      console.log(myUl.firstChild.innerText);
    }
  </script>

```



```

// <ul> 요소의 첫 번째 자식 노드의 HTML 내용을 콘솔에 출력
console.log(myUl.firstChild.innerHTML);
// <ul> 요소의 첫 번째 자식 노드의 부모 노드를 콘솔에 출력
console.log(myUl.firstChild.parentNode);

// "오렌지" <li> 요소를 변수 orange에 저장하고 콘솔에 출력
var orange = document.getElementsByTagName("li")[1];
console.log(orange.innerHTML);
// "오렌지" <li> 요소의 이전 형제 노드의 HTML 내용을 콘솔에 출력
console.log(orange.previousSibling.innerHTML);
// "오렌지" <li> 요소의 다음 형제 노드의 HTML 내용을 콘솔에 출력
console.log(orange.nextSibling.innerHTML);

// 새 <li> 요소를 생성하여 변수 newNode에 저장
var newNode = document.createElement("li");
// "수박" 텍스트 노드를 생성하여 변수 newText에 저장
var newText = document.createTextNode("수박");
// newText를 newNode의 자식으로 추가
newNode.appendChild(newText);
console.log(newNode);
// newNode를 <ul>의 자식으로 추가
myUl.appendChild(newNode);
// newNode를 "오렌지" <li> 요소의 앞에 추가 형제노드로 추가
myUl.insertBefore(newNode, orange);
// 기존의 newNode를 제거
myUl.removeChild(newNode);

// "수박"을 포함하는 새<li>요소를 생성하여 변수 newNode에 저장
var newNode = document.createElement("li");
newNode.innerHTML = "<b>수박</b>";
// newNode를 <ul>의 자식으로 추가
myUl.appendChild(newNode);
}
</script>
</head>
<body>
<div>
<p>과일리스트</p>
<ul>
<li>사과</li>
<li>오렌지</li>
<li>바나나</li>
</ul>
</div>
</body>
</html>

```

다음 이izzi 처럼 체크되어 있는 메뉴 버튼을 누를때 마다 아래 부분의 html를 변경하는 프로그램을 구현해 보자.



> 33. 정규식

정규식이란? 사용자의 입력이 원하는 입력인지 확인할 수 있는 문자 형태의 식을 의미한다. 과거에는 자바스크립트로 정규식 처리를 했는데 최근에는 HTML5에 정규식 처리 방법을 제공하고 있다. html5를 확인해 보고 이후 자바스크립트로 하는 방법을 확인해 볼 예정이다. 정규식의 경우 정규식 문법을 배워서 구현하면 많은 시간과 노력이 필요하다. 관련 문법은 나중에 자세히 공부하고 일단 챗GPT 한테 만들어 달라고 해서 사용하자

HTML5에서 제공하는 pattern 속성을 사용하면 사용자 입력 필드의 내용을 정규 표현식을 통해 유효성 검사할 수 있습니다. 이를 통해 사용자가 입력한 내용이 특정 패턴에 맞는지 브라우저에서 자동으로 확인할 수 있습니다.

예를 들어, 다음은 pattern 속성을 사용하여 4자에서 10자 사이의 알파벳과 숫자만 허용하는 입력 필드를 만드는 방법입니다:

```
<form>
  <input type="text" placeholder="4~10자리 입력" pattern="[A-Za-z0-9]{4,10}"
  maxlength="10" required>
  <button type="submit">확인</button>
</form>
```

여기서 주요 부분은 <input> 요소의 pattern 속성입니다. 위의 코드에서 사용한 패턴 [A-Za-z0-9]{4,10}은 다음과 같은 의미를 갖습니다:

[A-Za-z0-9]: 알파벳 대문자, 소문자, 숫자 중 하나의 문자를 의미합니다.

{4,10}: 앞의 패턴이 4에서 10회까지 반복되어야 함을 나타냅니다. 즉, 4자에서 10자 사이의 문자열을 의미합니다.

설정된 입력 필드는 사용자가 입력할 때 해당 패턴에 맞지 않는 경우 브라우저가 자동으로 경고 메시지를 표시하게 됩니다.

HTML5의 pattern 속성을 사용하면 클라이언트 측에서 간단한 입력 유효성 검사를 수행할 수 있으며, 사용자가 유효한 데이터를 입력하도록 도와줍니다. 패턴 속성 에 들어가는 문자열이 정규식에 해당한다.

다음 제공된 HTML 및 JavaScript 코드는 사용자 입력을 검증하기 위해 정규 표현식을 사용하는 방법을 보여줍니다.

비밀번호 검증 (<input id="reg2">):

비밀번호 검증을 위해 사용된 정규 표현식은 /^[A-Za-z0-9]{6,12}\$/입니다.

이 정규 표현식은 알파벳과 숫자만 포함하는 비밀번호를 확인합니다.

비밀번호는 6자에서 12자 사이여야 합니다.

"버튼"이 클릭될 때, JavaScript는 test()를 사용하여 입력이 정규 표현식과 일치하는지 확인합니다.

이메일 검증 (<input id="reg3">):

이메일 검증을 위해 사용된 정규 표현식은

/^[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*@[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*.[a-zA-Z]{2,3}\$/i입니다.

이 정규 표현식은 이메일 형식을 확인합니다.

"버튼"이 클릭될 때, JavaScript는 test()를 사용하여 입력이 정규 표현식과 일치하는지 확인합니다.

전화번호 검증 (<input id="reg4">):

전화번호 검증을 위해 사용된 정규 표현식은 /^\\d{3}-\\d{3,4}-\\d{4}\$/입니다.

이 정규 표현식은 전화번호 형식을 확인합니다.

"버튼"이 클릭될 때, JavaScript는 test()를 사용하여 입력이 정규 표현식과 일치하는지 확인합니다.

이 코드는 HTML5에서 제공하는 pattern 속성을 사용하여 입력 유효성 검사를 수행하는 부분도 포함하고 있습니다. 이를 통해 사용자가 입력한 내용이 특정 패턴에 부합하는지 확인할 수 있습니다.

정규표현식(Regular Expression)은 문자열 패턴을 검색하여 일치하는 문자를 찾는 데 사용됩니다. 주로 이메일, 주민번호, 전화번호 등과 같이 입력 데이터가 올바른 형식인지 확인할 때 활용됩니다. 정규식은 다음과 같은 형태로 표현됩니다:

// 정규식 표현 예시

/문자열패턴기술/추가정보

여기서 추가정보는 조합하여 사용할 수 있으며, 주로 다음과 같은 옵션이 사용됩니다:

i: 대소문자를 구분하지 않고 검색

g: 전체에서 여러 개의 일치하는 문자열 찾기

m: 줄당 시작하는 문자열이 같을 경우

정규식은 주로 문자열 검색, 치환, 패턴 일치 여부 확인 등에 사용됩니다. 각 메소드의 간단한 설명은 다음과 같습니다:

search: 문자열에서 일치하는 문자의 인덱스를 반환

replace: 문자열에서 일치하는 문자를 다른 값으로 변경

test: 문자열에 일치하는 패턴이 있으면 true, 없으면 false 반환

match: 문자열에서 일치하는 문자열들을 배열로 반환

예를 들어, 다음과 같은 형태로 사용할 수 있습니다:

```
var pattern = /정규식패턴/i; // 대소문자를 구분하지 않는 정규식 패턴
```

```
var searchString = "검색 대상 문자열";
```

```
var resultIndex = searchString.search(pattern);
```

```
var replacedString = searchString.replace(pattern, "대체 문자열");
```

```
var isPatternMatched = pattern.test(searchString);
```

```
var matchedStrings = searchString.match(pattern);
```

```
// 정규식 패턴이 시작하는 인덱스 (예: -1은 일치하는 패턴이 없음을 나타냄)
```

```
console.log(resultIndex);
```

```
// 검색 대상 문자열에서 정규식 패턴을 "대체 문자열"로 교체한 결과
```

```
console.log(replacedString);
```

```
// 검색 대상 문자열에 정규식 패턴이 일치하는지 여부 (true 또는 false)
```

```
console.log(isPatternMatched);
```

```
// 검색 대상 문자열에서 정규식 패턴에 일치하는 문자열들을 배열로 반환
```

```
console.log(matchedStrings);
```

이런 식으로 정규식을 활용하여 문자열 처리 작업을 효과적으로 수행할 수 있습니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script>

    var str="visit Human";
    var index = str.search("Human");
    console.log(index); // 문자열에서 Human이 시작되는 인덱스 6 리턴
    var str="visit Human";
    var index = str.search(/Human/);
    console.log(index);

    var str="visit Human";
    var index = str.search(/human/); //-1
    console.log(index);

    var str="visit Human";
    var index = str.search(/human/i); //6
    console.log(index);

    //replace
    var str="visit Human ";
    var result = str.replace(/human/i,"Microsoft");//6
    console.log(str); //visit Human
    console.log(result); //visit Microsoft

    var str="visit Human Human ";
    var result = str.replace(/human/i,"Microsoft");//6
    console.log(str); //visit Human Human
    console.log(result); //visit Microsoft Human

    var str="visit Human Human ";
    var result = str.replace(/human/ig,"Microsoft");//6
    console.log(str); //visit Human Human
    console.log(result); //visit Microsoft Microsoft

    //test
    var str="visit Human Human ";
    var pattern=/e/;
    console.log(pattern.test(str)); //e 가 없으므로 false가 생성
    console.log(/i/.test(str)); //i가 있으므로 true가 생성
    //정규표현식 패턴 /i/ 에서 i부분에 올수 있는 패턴
```

```

//[abc] abc문자중 하나의 문자를 검색한다.
var str="visit Human Human ";
var result = str.match(/[i]/g);
console.log(result); //i,i
var result = str.match(/[im]/g);
console.log(result); //i,i,m,m
var result = str.match(/[ims]/g);
console.log(result); //i,s,i,m,m
// "ab" 문자열을 검색
var result = str.match(/ab/g);
console.log(result); // null (문자열에 "ab"가 없음)

//[0-9] 0부터 9 사이에 있는 문자를 선택한다.
//[0123456789]와 같은 의미
var str="v0123456789";
var result = str.match(/[3-6]/g);
console.log(result); //3,4,5,6
//(xx|yy) xx이거나 yy일대 선택한다.
var str="visit Human Human ";
var result = str.match(/(visit|Human)/g);
console.log(result); //["visit", "Human", "Human"]

// /d 숫자만 찾는다.
var str="visit1 Human10 Human100 ";
var result = str.match(/\d/g);
console.log(result); //["1", "1", "0", "1", "0", "0"]

//다음 예제는 사용자 입력을 받아서 다양한 정규식을 확인하는 예제들이다.
var regExp = /^[A-Za-z0-9]{6,12}$/; //1. 숫자와 문자 포함 형태의 6~12자리 이내
var regExp =
/^[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*@[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*.[a-zA-Z]{2,
3}$/i; //2. 이메일 정규식
var regExp = /^\d{3}-\d{3,4}-\d{4}$/; //3. 핸드폰번호 정규식

window.onload=function(){
document.getElementById("b2").addEventListener("click",function(){
    var regExp = /^[A-Za-z0-9]{6,12}$/;
    if(regExp.test(document.getElementById("reg2").value)){
        alert("사용할 수 있는 아이디");
    }else{
        alert("사용할 수 없는 아이디");
    }
})

document.getElementById("b3").addEventListener("click",function(){
    var regExp =

```

```

/^[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*@[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*.[a-zA-Z]{2,
3}$/i;

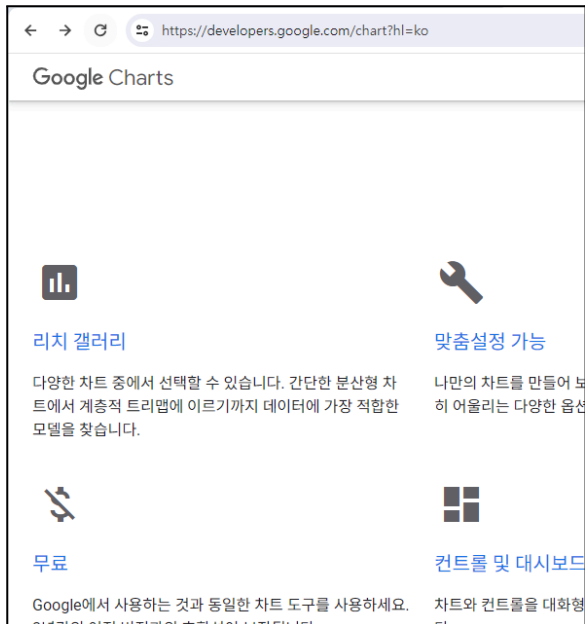
        if(regExp.test(document.getElementById("reg3").value)){
            alert("사용할 수 있는 이메일");
        }else{
            alert("사용할 수 없는 이메일");
        }
    })

document.getElementById("b4").addEventListener("click",function(){
    var regExp = /^\\d{3}-\\d{3,4}-\\d{4}$/;
    if(regExp.test(document.getElementById("reg4").value)){
        alert("사용할 수 있는 전화번호");
    }else{
        alert("사용할 수 없는 전화번호");
    }
})
}
</script>
</head>
<body>

    숫자와 문자 포함 형태의 6~12자리 이내의 암호 정규식
    <input type="text" id="reg2"><button id="b2">버튼</button><br>
    이메일 정규식
    <input type="text" id="reg3"><button id="b3">버튼</button><br>
    핸드폰번호 정규식
    <input type="text" id="reg4"><button id="b4">버튼</button><br>
    <br><br>
    html5 정규식
    <form>
        <input type="text" placeholder="4~10자리 입력"
pattern="[A-Za-z0-9]{4,10}" maxlength="10" required>
        <button type="submit">check</button>
    </form>
<!--
    placeholder 사용자 입력전 입력 가이드
    maxLength 사용자 입력 최대 크기
    required 사용자가 입력을 반드시 해야 한다.
-->
</body>
</html>

```

> 31. 구글차트 사용하기

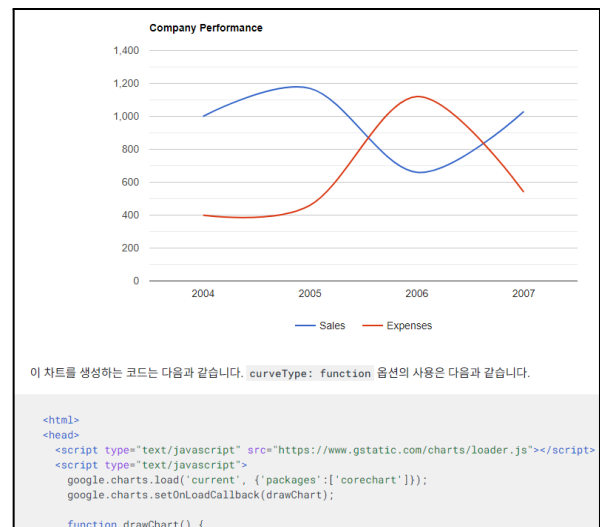
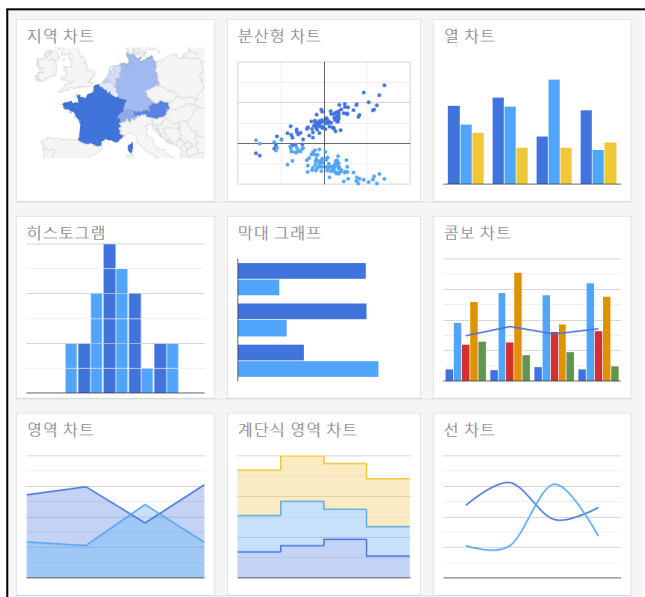


구글 차트를 이용해서 쉽게 원하는 차트를 만들 수 있다. 다음 사이트에 들어가 보자.

<https://developers.google.com/chart?hl=ko>

왼쪽 이미지 리치 갤러리를 클릭하면 다음 이미지 처럼 다양한 차트가 제공 된다.

본인이 원하는 차트 모양을 선택한다. 여기서는 선차트를 선택 하고 다음 아래 오른쪽 이미지 처럼 차트 아래 javascript 예제를 복사한후 html파일로 만들어 실행하면 원하는 차트를 만들 수 있다.



차트의 코드를 잘 확인해서 값을 변경해서 원하는 차트모양을 만들어 보자.

> 29. 미로찾기 프로젝트 1

이 코드는 미로 게임을 만드는 HTML과 JavaScript로 구성된 예제입니다. 이 코드를 통해 플레이어가 미로를 탐험하고 목표 지점에 도달 할 수 있습니다.

map 배열: 미로의 구조를 표현하는 배열로, 숫자로 구성되어 있습니다.

0: 빈 공간 1: 벽 2: 목표 지점 3: 플레이어의 현재 위치를 나타냅니다

madeTable(y, x): 테이블을 생성하는 함수입니다. y와 x는 테이블의 행과 열 수를 나타냅니다.

drawArray(arr): map 배열을 기반으로 테이블을 채우는 함수입니다.

isMove(y, x): 주어진 위치 (y, x)가 이동 가능한지 확인하는 함수로, 해당 위치의 값이 1이면 이동 불가능하고, 그 외의 값이면 이동 가능합니다.

unityY와 unitX: 플레이어의 현재 위치를 나타내는 변수입니다.

endY와 endX: 목표 지점의 위치를 나타내는 변수입니다.

window.onload 이벤트 핸들러:

window.onload 이벤트 핸들러는 HTML 문서가 완전히 불러 지고 나면 실행된다. 이 때 게임 초기화와 관련된 작업을 수행합니다.

document.body.innerHTML을 사용하여 HTML 문서의 내용을 현재 생성된 미로 테이블로 대체합니다.

drawArray(map)을 호출하여 map 배열에 따라 테이블을 칠합니다. 즉, 미로의 벽, 목표 지점 및 플레이어의 초기 위치를 그립니다.

document.body.onkeypress 이벤트 핸들러:

이벤트 핸들러는 사용자가 키보드 키를 눌렀을 때 실행됩니다. 키 코드에 따라 다양한 동작을 처리합니다.

event.keyCode를 사용하여 사용자가 어떤 키를 눌렀는지 확인합니다.

키 이벤트 처리 (switch(event.keyCode)):

스위치문을 사용하여 event.keyCode 값에 따라 다음과 같은 동작을 수행합니다.

56 (키패드의 8 키) - 위로 이동

54 (키패드의 6 키) - 오른쪽으로 이동

52 (키패드의 4 키) - 왼쪽으로 이동

50 (키패드의 2 키) - 아래로 이동

isMove 함수를 사용하여 플레이어가 이동 가능한 위치인지 확인하고, 가능하다면 현재 위치에 플레이어를 지우고 새로운 위치에 플레이어를 그립니다.

게임 종료 조건 확인:

플레이어의 현재 위치가 목표 지점의 위치와 일치하면 게임 종료 메시지를 표시하고 게임이 종료됩니다.

이 코드를 실행하면 플레이어가 키보드를 사용하여 미로를 탐험하고 목표 지점에 도달하는 미로 게임을 플레이할 수 있습니다. 게임 진행시 넘버키를 키고 해야 한다.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    var map=[
      [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
      [1,0,0,0,0,0,0,0,0,0,0,1,1,0,2],
      [1,0,1,1,1,1,1,1,1,1,0,0,1,0,1],
      [1,0,1,0,0,0,0,0,0,1,1,0,1,0,1],
      [1,0,1,0,1,1,1,1,0,1,0,0,1,0,1],
      [1,0,1,0,1,1,0,0,0,1,0,1,1,0,1],
      [1,0,1,0,1,0,0,1,0,1,0,0,1,0,1],
      [1,0,1,0,1,0,0,1,0,1,0,0,1,0,1],
      [1,0,1,0,1,0,1,0,1,0,1,0,1,0,1],
      [1,0,0,0,1,0,0,0,1,0,0,0,1,0,1],
      [1,1,1,1,1,0,1,0,1,1,0,1,1,0,1],
      [1,0,0,0,1,0,1,0,1,1,0,0,1,0,1],
      [1,0,1,0,1,0,1,0,0,1,1,0,1,0,1],
      [1,0,1,0,1,1,1,1,0,1,1,0,1,0,1],
      [1,3,1,0,0,0,0,0,0,1,0,0,0,0,1],
      [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
    ];
    function madeTable(y,x){
      var str="";
      str+="




```

```

        for(var i=0;i<y;i++){
            for(var j=0;j<x;j++){
                //str+=`<td id=y${i}x${j}>`;
                switch(arr[i][j]){
                    case 1:
document.getElementById("y"+i+"x"+j).setAttribute("bgcolor","red");
                    break;
                    case 2:
document.getElementById("y"+i+"x"+j).setAttribute("bgcolor","yellow");
                    break;
                    case 3:
document.getElementById("y"+i+"x"+j).innerHTML="<img src='img/k.jpg' width=20
height=20>"
                    break;
                }
            }
        }
    }
    function isMove(y,x){
        //이동 가능하면 true , false
        if(map[y][x]==1){
            return false;//이동 불가하면 false
        }else{
            return true;//이동 가능하면 true
        }
    }
    var unitY=13;
    var unitX=1;
    var endY=1;
    var endX=14;
    window.onload=function(){
        //테이블 만들기
        document.body.innerHTML=madeTable(map.length,map[0].length);
        //테이블 칠하기
        drawArray(map);
        //키보드 이벤트 처리
        document.body.onkeypress=function(){
            //alert(event.keyCode);
            //up 56 right 54 left 52 down 50
            //이동하려면 현재위치에 unit를 지운다.
            //이동할 위치를 계산한다. 위로이동하려면 y=y-1
            //이동할 위치에 unit를 그린다.
            //switch값으로 4개중 하나를 확인해서 이동시킨다.
            switch(event.keyCode){
                case 56://up
                    if(isMove(unitY-1,unitX)){

```

```

document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
    unitY=unitY-1;
document.getElementById(`y${unitY}x${unitX}`).innerHTML=
    "<img src='img/k.jpg' width=20 height=20>";
    }
    break;
    case 54://right
        if(isMove(unitY,unitX+1)){
document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
            unitX=unitX+1;
document.getElementById(`y${unitY}x${unitX}`).innerHTML=
            "<img src='img/k.jpg' width=20 height=20>";
        }
        break;
    case 52://left
        if(isMove(unitY,unitX-1)){
document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
            unitX=unitX-1;
document.getElementById(`y${unitY}x${unitX}`).innerHTML=
            "<img src='img/k.jpg' width=20 height=20>";
        }
        break;
    case 50://down
        if(isMove(unitY+1,unitX)){
document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
            unitY=unitY+1;
document.getElementById(`y${unitY}x${unitX}`).innerHTML=
            "<img src='img/k.jpg' width=20 height=20>";
        }
        break;
    }
    //게임 종료 조건
    if(unitX==endX&&unitY==endY){
        alert("게임종료");
    }
}
}
</script>
</head>
<body>
</body>
</html>

```

> 30. 미로찾기 프로젝트 2 (리플레이)

이전 코드와 현재 코드 간의 주요 차이점은 리플레이(replay) 기능이 추가되었습니다. 리플레이 기능을 구현하기 위해 배열(replayX 및 replayY)을 사용하고, 리플레이 타이머(replayTimer)를 설정하여 플레이어의 움직임을 재생하는 부분이 추가되었습니다. 아래에서 주요 차이점을 설명합니다.

리플레이 배열(replayX 및 replayY):

replayX와 replayY 배열은 플레이어의 움직임을 저장합니다. 이동할 때마다 현재 위치의 x 좌표와 y 좌표를 해당 배열에 추가합니다.

리플레이 기능을 위해 각 움직임 단계의 좌표가 기록되며, 게임 종료 시 이 배열을 사용하여 움직임을 재생합니다.

replayFunction 함수:

replayFunction 함수는 리플레이 기능을 수행합니다. 현재 플레이어의 위치를 초기화하고, 리플레이 배열에서 다음 위치를 가져와 해당 위치에 플레이어 이미지를 그립니다.

리플레이 배열에서 위치를 하나씩 가져오고 배열에서 해당 위치를 제거하여 이동 순서대로 움직입니다.

리플레이 배열이 빈 경우(모든 움직임이 재생되면) 리플레이 타이머를 중지하고 게임을 종료합니다.

리플레이 타이머(replayTimer):

replayTimer 변수는 setInterval 함수를 사용하여 리플레이 함수(replayFunction)를 일정 시간 간격으로 호출합니다. 이를 통해 움직임이 자동으로 재생됩니다.

리플레이 시작 지점 기록:

게임 시작 시 replayX와 replayY 배열에 초기 플레이어 위치인 (1, 13) 좌표를 추가하여 리플레이 시작 지점을 기록합니다. 게임 진행중 유닛이 이동할때 마다 해당 좌표를 저장한다.

게임 종료 시 리플레이 시작:

게임이 종료되면 리플레이를 시작하고, 움직임이 자동으로 재생됩니다. 이를 통해 사용자가 플레이한 경로가 자동으로 재생됩니다.

이러한 변경으로 사용자는 게임을 플레이하고, 게임이 종료되면 리플레이를 시작하여 최근에 플레이한 경로를 다시 볼 수 있습니다.

```
<html lang="en">
```

```
<head>
```

```
  <script>
```

```
    var map=[
```

```
      [1,1,1,1,1,1,1,1,1,1,1,1,1,1],
```

```
      [1,0,0,0,0,0,0,0,0,0,0,1,1,0,2],
```

```
      [1,0,1,1,1,1,1,1,1,1,0,0,1,0,1],
```

```
      [1,0,1,0,0,0,0,0,0,1,1,0,1,0,1],
```

```
      [1,0,1,0,1,1,1,1,0,1,0,0,1,0,1],
```

```

[1,0,1,0,1,1,0,0,0,1,0,1,1,0,1],
[1,0,1,0,1,0,0,1,0,1,0,0,1,0,1],
[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1],
[1,0,0,0,1,0,0,0,1,0,0,0,1,0,1],
[1,1,1,1,1,0,1,0,1,1,0,1,1,0,1],
[1,0,0,0,1,0,1,0,1,1,0,0,1,0,1],
[1,0,1,0,1,0,1,0,0,1,1,0,1,0,1],
[1,0,1,0,1,1,1,1,0,1,1,0,1,0,1],
[1,3,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
];
function madeTable(y,x){
    var str="";
    str+="

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| ";             str+="</td>";         }         str+="</tr>";     }     str+="</table>";     return str; } function drawArray(arr){     var y=arr.length;     var x=arr[0].length;     for(var i=0;i<y;i++){         for(var j=0;j<x;j++){             //str+=" <td 2:="" 3:="" <="" break;="" case="" document.getelementbyid("y"+i+"x"+j).innerhtml="&lt;img src='img/k.jpg' width=20 height=20&gt;" document.getelementbyid("y"+i+"x"+j).setattribute("bgcolor","red");="" document.getelementbyid("y"+i+"x"+j).setattribute("bgcolor","yellow");="" id='y\${i}x\${j}&gt;";' pre="" switch(arr[i][j]){=""> </td> |  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|


```

```

        break;
    }
}
}
}
function isMove(y,x){
    //이동 가능하면 true , false
    if(map[y][x]==1){
        return false;//이동 불가하면 false
    }else{
        return true;//이동 가능하면 true
    }
}
function replayFunction(){
    document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
    unitX=replayX.shift();
    unitY=replayY.shift();
    document.getElementById(`y${unitY}x${unitX}`).innerHTML
    ="<img src='img/k.jpg' width=20 height=20>";
    if(replayX.length==0){
        clearInterval(replayTimer);                //타이머 종료
        alert("replay종료");                        //최단경로
    }
    //얕은 복사
    // var a=[1,2,3];
    //     var b=a;
    //     b[1]=100;
    //     console.log(a);
    //     console.log(b);
    //     //깊은 복사
    //     var a=[1,2,3];
    //     var b=a.slice();
    //     b[1]=100;
    //     console.log(a);
    //     console.log(b);
}
var replayTimer;
var unitY=13;
var unitX=1;
var endY=1;

```

```

var endX=14;
var replayX=[];
var replayY=[];
window.onload=function(){
    //replay배열에 유닛의 시작위치 저장
    replayX.push(unitX);
    replayY.push(unitY);
    //테이블 만들기
    document.body.innerHTML=madeTable(map.length,map[0].length);
    //테이블 칠하기
    drawArray(map);
    //키보드 이벤트 처리
    document.body.onkeypress=function(){
        //alert(event.keyCode);
        //up 56 right 54 left 52 down 50
        //이동하려면 현재위치에 unit를 지운다.
        //이동할 위치를 계산한다. 위로이동하려면 y=y-1
        //이동할 위치에 unit를 그린다.
        //switch값으로 4개중 하나를 확인해서 이동시킨다.
        switch(event.keyCode){
            case 56://up
                if(isMove(unitY-1,unitX)){
document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
                    unitY=unitY-1;

document.getElementById(`y${unitY}x${unitX}`).innerHTML=
                        "<img src='img/k.jpg' width=20 height=20>";
                }
                break;
            case 54://right
                if(isMove(unitY,unitX+1)){
document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
                    unitX=unitX+1;

document.getElementById(`y${unitY}x${unitX}`).innerHTML=
                        "<img src='img/k.jpg' width=20 height=20>";
                }
                break;
            case 52://left
                if(isMove(unitY,unitX-1)){

```



```

document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
        unitX=unitX-1;

document.getElementById(`y${unitY}x${unitX}`).innerHTML=
        "<img src='img/k.jpg' width=20 height=20>";
    }
    break;
    case 50://down
        if(isMove(unitY+1,unitX)){
document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
            unitY=unitY+1;
document.getElementById(`y${unitY}x${unitX}`).innerHTML=
            "<img src='img/k.jpg' width=20 height=20>";
        }
        break;
    }
    replayX.push(unitX);
    replayY.push(unitY);
    // var replay=[];
    // replayT={};
    // replayT.x=unitX;
    // replayT.y=unitY;
    // replay.unshift(replayT);
    //게임 종료 조건
    if(unitX==endX&&unitY==endY){
        alert("게임종료");
        console.log(replayX);
        replayTimer=setInterval(replayFunction,500);
    }
    }
}
</script>
</head>
<body>
</body>
</html>

```

> 31. 미로찾기 프로젝트 3 (최단거리)

코드를 비교하면서 변경 사항을 설명해드리겠습니다. 아래는 이전 코드와 현재 코드 간의 주요

차이점을 설명한 것입니다.

shortFunction 함수:

shortFunction 함수는 최단 경로 재생을 위한 함수입니다. 현재 플레이어의 위치를 초기화하고, shortX와 shortY 배열에서 다음 위치를 가져와 해당 위치에 플레이어 이미지를 그립니다.

shortX와 shortY 배열은 최단 경로 재생에 사용됩니다.

최단 경로 재생이 완료되면 해당 타이머를 중지하고 알림을 표시합니다.

tempX와 tempY 배열:

tempX와 tempY 배열은 replayX와 replayY 배열의 복사본으로 사용됩니다.

복사본 배열을 사용하여 최단 경로 계산을 수행합니다. 계산 중에 복사본 배열에서 값을 제거하며, 중복되는 좌표가 없을 때만 shortX와 shortY 배열에 값을 추가합니다.

중복되는 좌표가 발견되면 해당 좌표까지의 모든 좌표를 복사본 배열에서 제거합니다.

최단 경로 타이머(shortTimer):

shortTimer 변수는 setInterval 함수를 사용하여 shortFunction 함수를 일정 시간 간격으로 호출합니다.

최단 경로가 자동으로 재생됩니다.

최단 경로 계산 및 재생:

게임이 종료되면 최단 경로를 계산하고, 해당 경로를 replayX와 replayY 배열에서 중복되지 않도록 shortX와 shortY 배열에 저장합니다.

이렇게 저장한 최단 경로를 shortFunction 함수와 shortTimer를 사용하여 재생합니다.

이제 게임이 종료되면 최단 경로가 계산되어 자동으로 재생되고, 이 과정에서 중복되지 않는 경로가 shortX와 shortY 배열에 저장됩니다. 이를 통해 게임 경로의 최단 경로를 효과적으로 찾고 재생할 수 있습니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script>
    var map=[
      [1,1,1,1,1,1,1,1,1,1,1,1,1,1],
      [1,0,0,0,0,0,0,0,0,0,1,1,0,2],
      [1,0,1,1,1,1,1,1,1,1,0,0,1,0,1],
      [1,0,1,0,0,0,0,0,0,1,1,0,1,0,1],
      [1,0,1,0,1,1,1,1,0,1,0,0,1,0,1],
```

```

[1,0,1,0,1,1,0,0,0,1,0,1,1,0,1],
[1,0,1,0,1,0,0,1,0,1,0,0,1,0,1],
[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1],
[1,0,0,0,1,0,0,0,1,0,0,0,1,0,1],
[1,1,1,1,1,0,1,0,1,1,0,1,1,0,1],
[1,0,0,0,1,0,1,0,1,1,0,0,1,0,1],
[1,0,1,0,1,0,1,0,0,1,1,0,1,0,1],
[1,0,1,0,1,1,1,1,0,1,1,0,1,0,1],
[1,3,0,0,0,0,0,0,0,0,0,0,0,0,1],
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
];

function madeTable(y,x){
    var str="";
    str+="

||
||
||


```

```

        //str+=`<td id=y${i}x${j}>`;
        switch(arr[i][j]){
            case 1:
document.getElementById("y"+i+"x"+j).setAttribute("bgcolor","red");
                break;
            case 2:
document.getElementById("y"+i+"x"+j).setAttribute("bgcolor","yellow");
                break;
            case 3:
document.getElementById("y"+i+"x"+j).innerHTML=`<img
src='img/k.jpg' width=20 height=20>`
                break;
        }
    }
}
}

function isMove(y,x){
    //이동 가능하면 true , false
    if(map[y][x]==1){
        return false;//이동 불가하면 false
    }else{
        return true;//이동 가능하면 true
    }
}

function shortFunction(){
    document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
    unitX=shortX.shift();
    unitY=shortY.shift();
    document.getElementById(`y${unitY}x${unitX}`).innerHTML
    =`<img src='img/k.jpg' width=20 height=20>`;
    if(shortX.length==0){
        clearInterval(shortTimer);
        alert('최단경로 완료');
    }
}

function replayFunction(){
    document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
    unitX=replayX.shift();

```

```

unitY=replayY.shift();
document.getElementById(`y${unitY}x${unitX}`).innerHTML
="";
if(replayX.length==0){
    //타이머 종료
    clearInterval(replayTimer);
    alert("replay종료");
    // tempX=replayX.slice();
    // tempY=replayY.slice();

    //tempX.shift(),tempY.shift()의 값과 동일한값이 배열에 없으면 출력
    //특정값이 존재하는지 확인

    //tempX.includes(x);
    while(tempX.length!=0){
        var x=tempX.shift();
        var y=tempY.shift();
        if(!isUnitXY(x,y)){
            shortX.push(x);
            shortY.push(y);
        }else{
            while(!(x==tempX[0]&&y==tempY[0])){
                tempX.shift();
                tempY.shift();
            }
        }
        console.log(shortX);
        console.log(shortY);
    }

    shortTimer=setInterval(shortFunction,250);
    //타이머를 이용해서 최단경로를 replay해보자.
    //최단경로
}
}

var shortTimer;
var replayTimer;
var unitY=13;
var unitX=1;

```

```

var endY=1;
var endX=14;

var replayX=[];
var replayY=[];
var tempX=[];
var tempY=[];
var shortX=[];
var shortY=[];

window.onload=function(){
    //replay배열에 유닛의 시작위치 저장
    replayX.push(unitX);
    replayY.push(unitY);

    //테이블 만들기
    document.body.innerHTML=madeTable(map.length,map[0].length);
    //테이블 칠하기
    drawArray(map);
    //키보드 이벤트 처리
    document.body.onkeypress=function(){
        //alert(event.keyCode);
        //up 56 right 54 left 52 down 50
        //이동하려면 현재위치에 unit를 지운다.
        //이동할 위치를 계산한다. 위로이동하려면 y=y-1
        //이동할 위치에 unit를 그린다.
        //switch값으로 4개중 하나를 확인해서 이동시킨다.
        switch(event.keyCode){
            case 56://up
                if(isMove(unitY-1,unitX)){
document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
                    unitY=unitY-1;

document.getElementById(`y${unitY}x${unitX}`).innerHTML=
                        "<img src='img/k.jpg' width=20 height=20>";
                }
                break;
            case 54://right
                if(isMove(unitY,unitX+1)){
document.getElementById(`y${unitY}x${unitX}`).innerHTML="";

```

```

        unitX=unitX+1;

document.getElementById(`y${unitY}x${unitX}`).innerHTML=
    "<img src='img/k.jpg' width=20 height=20>";
    }
    break;
    case 52://left
        if(isMove(unitY,unitX-1)){
document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
            unitX=unitX-1;

document.getElementById(`y${unitY}x${unitX}`).innerHTML=
                "<img src='img/k.jpg' width=20 height=20>";
            }
            break;
            case 50://down
                if(isMove(unitY+1,unitX)){
document.getElementById(`y${unitY}x${unitX}`).innerHTML="";
                    unitY=unitY+1;

document.getElementById(`y${unitY}x${unitX}`).innerHTML=
                        "<img src='img/k.jpg' width=20 height=20>";
                    }
                    break;
                }
                replayX.push(unitX);
                replayY.push(unitY);
                // var replay=[];
                // replayT={};
                // replayT.x=unitX;
                // replayT.y=unitY;
                // replay.unshift(replayT);
                //게임 종료 조건
                if(unitX==endX&&unitY==endY){
                    alert("게임종료");
                    tempX=replayX.slice();
                    tempY=replayY.slice();

                    console.log(replayX);
                    replayTimer=setInterval(replayFunction,250);
                }

```

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

> 32. 미로찾기 프로젝트 4 (자동길찾기)

미로에서 벽을 집고 움직이면 언젠가는 출구에 도착한다. 이 알고리즘을 기준으로 프로그램을 구현하면 유닛의 진행방향을 저장할 수 있는 변수가 추가로 필요하고 오른쪽 벽에 붙어 있는지 확인하기 위해서 손이 없으므로 오른쪽으로 회전 한 다음 막혀있으면 벽 아니면 벽이 아닌 것이 된다.

이를 바탕으로 한칸 이동할 때마다 다음과 같은 로직을 사용하게 된다.

```
right();//벽인지 확인하기 위해서 오른쪽 회전
while(!isMove()){// 이동가능한가?
    left();//이동 불가능하면 오른쪽 벽에 붙어 있으므로 이제는 이동 가능할때 까지 왼쪽으로
    회전 하면된다.
}
go();//전진한다.
```

한칸이동하는 로직을 도착 할때까지 무한 반복하면된다.

```
while(!isEnd()){//종료 되었는가?
    right();
    while(!isMove()){
        left();
    }
    go();//전진한다.
}
```

알고리즘을 이해하는 것은 힘든 일이다. 이렇게 하면된다 하고 넘어가면 된다.

최단거리를 구한후 리플레이하면 미로를 최단 거리로 이동하는 것을 확인할 수 있다.