Javascript

Ⅰ. 개발환경 설정과 시작하기 전에

1. Visual Studio Code 설치

https://code.visualstudio.com/

2. Visual Studio Code Extensions 설치

- HTML Snippets

- HTML CSS Support

- JavaScript (ES6) code snippets

- Live Server

- Open In Default Browser

~~3. 에디터 배경색 바꾸기~~

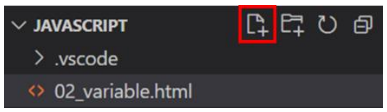
파일 > 기본설정 > 워크벤치 > 설정검색 : color customizations 로 검색 > Editor:Token Color Customizations의 "settings.json에서 편집"

아래와 같이 "editor.background" :"#efefef" 추가

```
{
    "workbench.colorCustomizations": {    ➔ 기존내용
        "editor.background" :"#efefef"      ➔ 추가내용
    }
}
```

2. 기본 코딩 작성법

1) HTML5 형식의 문서 생성



① 왼쪽 그림과 같이 탐색기 패널의 폴더 이름 오른쪽의 "새파일" 아이콘을 선택한 후 확장자가 html인 새파일을 생성

② 편집창에 느낌표(!) 하나를 입력하고 엔터

③ <html lang="en">을 <html lang="ko">로 수정

2) 자바스크립트 기본 형식

① HTML문서안에 삽입

```
<head>
    <script language="javascript" src="js파일의 경로 및 파일명">
        자바스크립트 코드
    </script>
</head>
```

② 외부 파일 로드

```
<head>
    <script src="js파일의 경로 및 파일명"></script>
</head>
```

③ 태그에 직접 작업

```
<input type="button" onclick="console.log('Hello. Javascript!!!');">
```

Ⅱ. Javascript Basic

1. 변수 (Variable)

어렵게 생각하지 말자. 변수는 로컬 컴퓨터의 메모리 상에 작성된 Javascript가 실행되는 동안 임시로 데이터를 저장하고 있는 이름이 주어진 메모리 공간이라고 생각하면 된다. 이때 저장된 데이터는 항상 마지막에 저장된 데이터만 유지된다.

1) 변수의 선언

기본적으로 Javascript에서는 변수를 선언하지 않고 바로 사용도 가능하다. 다만, 함수와 함께 사용이 될 경우 변수를 선언한 것과 선언하지 않고 사용할 경우엔 차이가 발생한다.

```
var 변수명; //변수를 선언만 한 경우
var 변수명2=data; //변수의 선언과 초기값(data)를 함께 지정하여 선언한 경우
```

2) 전역변수

- HTML문서가 브라우저에 로딩되어 있는 동안 변수의 데이터가 유지됨.
- HTML문서 내 어디서든 변수를 사용 가능함.

3) 지역변수

- 사용자 정의 함수 내에서만 데이터가 유지되며 함수가 종료되면 변수의 값을 상실함.
- 함수 바깥에선 지역변수를 사용 할 수 없음

.

2. 데이터 타입

데이터 타입이란 자료(data)의 형식을 의미하는 것으로, 문자, 문자열, 숫자, 참(true)과 거짓(false) 등이 대표적인 데이터 타입이다.
다만, Javascript에서는 변수 선언에 있어 데이터 타입을 지정하지 않으며, 변수에 서로 다른 데이터 타입의 데이터를 저장 할 수도 있다.

1) 개발에 있어 알아두면 도움이 되는 데이터 타입

- null : 일반적으로 선언이 되지 않은 변수의 반환 값.

```
<script>
    console.log(test); //null 로 error
</script>
```

위 test는 변수의 선언이 되지 않아 null로 출력의 대상이 되지 않아 error.

- undefined : 변수의 선언은 되었으나 값이 할당되지 않은 경우.

```
<script>
    var test;
```

```
    console.log(test); //undefined 출력
</script>
```

위 test는 변수로 선언이 되었으나 값이 할당되지 않아 undefined를 출력함.

3. 연산자

1) 산술연산자

```
+, -, *, /, %
```

```
<script>
        console.log(5+3); //덧셈
        console.log(5-3); //뺄셈
        console.log(5*3); //곱셈
        console.log(5/3); //나눗셈
        console.log(5%3); //나머지
    </script>
```

[질문]

```
<script>
    console.log(5*4/2); //result : ?
    console.log(5/4*2); // result : ?
    console.log(5/2%2); // result : ?
    console.log(5%3*2); // result : ?
</script>
```

2) 산술 연산자(증감 연산자)

```
++, --
```

[질문]

```
<script>
    var num=5;
    console.log(num++); //result : 출력(?), num(?)
    console.log(++num); //result : 출력(?), num(?)
    console.log(num--);  //result : 출력(?), num(?)
    console.log(--num);  //result : 출력(?), num(?)
</script>
```

3) 할당 연산자

=, =+, =-, =*, =/, =%

[질문]

```
<script>
    var num;
    console.log(num=5);             //result : 출력(?), num(?)
    console.log(num+=5);            //result : 출력(?), num(?)
    console.log(num-=2);            //result : 출력(?), num(?)
    console.log(num*=3);            //result : 출력(?), num(?)
    console.log(num/=2);            //result : 출력(?), num(?)
    console.log(num%=5);            //result : 출력(?), num(?)
</script>
```

4) (문자열)연결 연산자 : +

```
<script>
    var num=10;
    //연결 연산자는 문자열과 숫자를 순서대로 나열한다.
    console.log("홍길동은 " + num + "살이다.");
    console.log("홍길동은 " + num + 5 + "살이다.");
    console.log(num + 10 + "개의 사과");
    console.log("홍길동은 " + (num + 5) + "살이다.");
</script>
```

[질문]

```
<script>
    var num=10;
    //연결 연산자는 문자(열)과 숫자를 순서대로 나열한다.
    console.log("홍길동은 " + num + "살이다.");
    console.log("홍길동은 " + num + 5 + "살이다."); //result : ?

    console.log(num + 10 + "개의 사과"); //result : ?
    console.log("홍길동은 " + (num + 5) + "살이다."); //result : ?
</script>
```

5) 비교 연산자

==, !=, ~~===, !==,~~ <, <=, >, >=

```
<script>
```

```
    var num=10;
    console.log(num==10);//result : ?
    console.log(num!=10);              //result : ?
    console.log(num===10);             //result : ?
    console.log(num==="10");           //result : ?
    console.log(num!=="10");           //result : ?
</script>
```

6) 논리 연산자

||, &&, !

```
    var num=10;
    var sum; //선언만 하고 값을 할당하지 않음(undefined)
    console.log(true || false || false); //하나 이상이 true 이면 결과값은 true
    console.log(true && true && false); //모두 true 여야만 true
    console.log(true && !(true && false)); //값을 반대로
```

[질문]

```
    var num=10;
    var sum; //선언만 하고 값을 할당하지 않음(undefined)
    console.log(true || false || false); //하나 이상이 true 이면 결과값은 true
    console.log(true && true && false); //모두 true 여야만 true
    console.log( true && !(true && false) ); //값을 반대로
    console.log(true || false && true); //result : ?
    console.log(true && false || true); //result : ?
    console.log(sum); //result : ?
    console.log( !sum ); //result : ?
```

4. 함수

특별한 목적의 작업을 수행하도록 설계된 독립적인 블록으로 그 내부에 변수를 선언할 수 있으며, 매개변수로 값을 전달 할 수 있다.
또한 함수를 호출(실행)한 곳으로 특정 값(data)를 반환 할 수 있다.

1) 함수 선언문

```
function 함수명 ( [매개변수1, 매개변수 2 ... ( {
    실행문;
    [ return 값; ]
}
```

```
const 함수명 = function ( [매개변수1, 매개변수 2 ... ( {
    실행문;
    [ return 값; ]
}
```

```
function cmd_1() {
    console.log("함수 선언 1");
}
const cmd_2 = function () {
    var sum;
    var num=5;
    sum = num + 7;
    console.log("num : " + num + ", sum : " + sum);
}
cmd_1(); //함수 cmd-1 호출
cmd_2(); //함수 cmd-1 호출
```

2) 지역변수와 전역변수

① 전역변수와 지역변수

전역변수는 함수 바깥쪽에서 선언된 변수로 모든 함수에서 사용이 가능하며, 변수의 데이터는 최종적으로 저장된 데이터가 문서를 벗어나기 이전가지 유지된다.

지역변수는 선언된 함수 안에서만 사용이 가능하여 변수의 생명은 함수 시작부터 종료 까지다.

함수 내에서 선언되지 않은 변수는 전역변수로 보면 된다.

```
var sum=0;
const cmd_1 = function() {
    var num=5;
    sum = num + 7;
```

```
    console.log("num : " + num + ", sum : " + sum);
}
const cmd_2 = function() {
    console.log("sum : " + sum); //sum(?)
}
cmd_1(); //함수 cmd-1 호출
cmd_2(); //함수 cmd-1 호출
//console.log("num : " + num + ", sum : " + sum); //num(?), sum(?)
```

② 함수내에서 선언되지 않은 변수 : 전역변수

```
//var sum=0;
const cmd_1 = function() {
    var num=5;
    sum = num + 7;
    console.log("num : " + num + ", sum : " + sum);
}
const cmd_2 = function() {
    console.log("sum : " + sum); //sum(?)
}
cmd_1(); //함수 cmd_1 호출
cmd_2(); //함수 cmd_1 호출
</script>
```

③ 전역변수와 동일한 이름의 지역변수

```
    var sum=0;
    const cmd_1 = function() {
        var num=5;
        var sum=10;
        sum = num + 7;
      //parent.sum = num + 7;

        console.log("num : " + num + ", sum : " + sum);
    }
    const cmd_2 = function() {
        console.log("sum : " + sum); //sum(?)
    }
    cmd_1(); //함수 cmd_1 호출
    cmd_2(); //함수 cmd_1 호출
```

3) 매개변수

매개변수는 함수의 외부에서 보내오는 값(데이터)을 받을 수 있는 함수내에서 사용 되는 임시 메모리 공간이며, 함수 내에서는 지역변수처럼 사용이 가능하다.

```
<script>
    var sum=0;
    const cmd_1 = function(n) {
        sum = n + 5;
        cmd_2(5); //함수 cmd-1 호출
    }
    const cmd_2 = function(num) {
        sum += num;
        console.log("num : " + num + ", sum : " + sum);
    }
    cmd_1(10); //함수 cmd-1 호출
</script>
```

4) 함수 값의 반환

함수내에서 만들어진 값을 함수를 호출한 곳으로 반환하여 전역변수의 사용을 줄일 수 있음.

```
<script>
    var cmd1;
    const cmd_1 = function(n) {
        var sum;
        sum = n + 5;
        return sum;
    }
     const cmd_2 = function(num) {
        var sum=10;
        sum += num;
        console.log("num : " + num + ", sum : " + sum);
    }
    num1=cmd_1(10); //함수 cmd-cmd-1 호출
    cmd_2(num1)
</script>
```

5. 제어문

제어문은 조건을 기준으로 블록"{ }"내의 문장을 실행할지 여부를 결정하는 문장을 의미하며, 크게 제어문, 분기문, 반복문으로 구분되어 진다.

1) if()문

① 형식 1

```
if(조건) {
    블록문;
}
```

```
<script>
    var sum=0;
    var kor=70;
    var eng=80;
    var mat=30;
    var msg="";
    sum=kor + eng + mat;
    avg = sum/3;
    //평균이 80 점 이상이면 우수
    if(avg>=80) {
        msg="우수";
    }
    console.log(msg + "한 학생입니다.");
    </script>
```

[실습 예제 -1]

1. 위 예제를 수정하여 아래와 같은 결과를 출력할 수 있도록 한다.
2. if()문 1형식을 사용하고, 변수의 수와 이름은 변경하지 않는다.
2. 평균 점수가 80점 이상이면 "성적이 우수한 학생입니다."를 출력한다.
3. 평균 점수가 80점 미만이면 "성적이 저조한 학생입니다."를 출력한다.

```
<script>
    var sum=0;
    var kor=70;
    var eng=80;
    var mat=30;
```

```
    var msg="성적이 저조한 학생입니다.";
    sum=kor + eng + mat;
    avg = sum/3;
    //평균이 80 점 이상이면 우수
    if(avg>=80) {
        msg="성적이 우수한 학생입니다.";
    }
    console.log(msg);
</script>
```

[자율 실습 예제 : 511_control_11-ex.html]

1. 국어, 영어, 수학 점수를 지정
2. 세 과목의 합과 평균을 구하고, 이를 변수에 저장
3. 학생이름을 변수에 저장
4. 학생의 성별을 숫자로 저장
5. 평균 점수가 80점 이상이면 아래와 같이 출력하고 그렇지 않으면 아무것도 출력하지 않는다.
   ➔ 남학생 ○○○님은 우수한 학생입니다.

```
    <script>
        var sum=0;
        var kor=70;
        var eng=80;
        var mat=90;
        var sex = 1;//남자
        var stdName = "우영우";
        var msg="";
        sum=kor + eng + mat;
        avg = sum/3;
        //평균이 80 점 이상이면 우수
        if(avg>=80 && sex==1) {
            msg="남학생 " + stdName + "는 성적이 우수한 학생입니다.";
            console.log(msg);
        }
    </script>
```

② 형식 2

```
if(조건) {
    블록문 1;
}
else {
    블록문 2;
}
```

[실습예제 : 512_control_1.html]

```
<script>
    //홀수와 짝수 구별 함수
    function cmdFun(num){
        if(num%2==0) {
            console.log(num+"은 짝수입니다.");
        }
        else {
            console.log(num+"은 홀수입니다.");
        }
    }
    //함수 호출
    cmdFun(3);
</script>
```

[응용예제 : 512_control_ex.html]

1. "실습 예제 - 1"의 msg 변수는 사용하지 않는다.
2. if{}문 형식 2를 이용하여 "실습 예제 -1"과 동일한 결과가 나올 수 있도록 코드를 수정 한다.

```
<script>
    var sum=0;
    var kor=70;
    var eng=80;
    var mat=30;
    sum=kor + eng + mat;
    avg = sum/3;
    //평균이 80 점 이상이면 우수
    if(avg>=80) {
        console.log("성적이 우수한 학생입니다.");
    }
    else {
```

```
        console.log("성적이 저조한 학생입니다.");
    }
</script>
```

[자율실습예제 : 512_control-pre-ex.html]

1. 전역변수는 사용하지 않는다.
2. 국어, 영어, 수학 점수의 총점, 평균을 구한다.
3. 학생의 이름과 성별(남자-1, 여자-2)를 지정한다.
4. 남학생 중
   - 평균 점수가 80점 이상인 학생은 "남학생 ○○○님은 성적이 우수한 학생입니다."로 출력하고
   - 평균 점수가 80점 미만인 학생은 "남학생 ○○○님은 성적이 저조한 학생입니다."로 출력한다
5. 여학생은
   - 평균 점수가 80점 이상인 학생은 "여학생 ○○○님은 성적이 우수한 학생입니다."로 출력하고
   - 평균 점수가 80점 미만인 학생은 "여학생 ○○○님은 성적이 저조한 학생입니다."로 출력한다

```
<script>
    function cmdFun(mstdName, msex, mkor, meng, mmat) {
        var kor=mkor;
        var eng=meng;
        var mat=mmat;
        var stdName=mstdName;
        var sex=msex;
        var msg="";
        sum=kor + eng + mat;
        avg = sum/3;
        //남녀 구분
        if(sex==1) {
            if(avg>=80) {
                msg="남학생 " + stdName + "는 성적이 우수한 학생입니다.";
            }
            else {
                msg="남학생 " + stdName + "는 성적이 저조한 학생입니다.";
            }
        }
        else {
            if(avg>=80) {
                msg="여학생 " + stdName + "는 성적이 우수한 학생입니다.";
```

```
            }
            else {
                msg="여학생 " + stdName + "는 성적이 저조한 학생입니다.";
            }
        }
        console.log(msg);
    }
    //함수 호출
    cmdFun("우영우", 1, 0, 80, 70);
</script>
```

③ 형식 3

```
if(조건1) {
    블록문 1;
}
else if(조건 2) {
     블록문 2;
}
  :
  :
else {
    블록문 3;
}
```

[실습예제 : 513_control.html]

```
<script>
    //80 점 이상 : 우수한 점수입니다.
    //60 점 이상 : 평균적인 점수입니다.
    //60 점 미만 : 성적이 저조합니다.
    var num=70;
    if(num>=80) console.log("우수한 성적입니다.")
    else if(num>=60 && num<80) console.log("평균 성적입니다.");
    else console.log("저조한 성적입니다.");
</script>
```

[응용예제 : 513_control_ex.html]

1. 실습 예제를 참고한다.
2. 국어, 영어, 수학 점수를 입력 받아 총점과 평균을 구한다.
3. 평균을 이용하여 "이번 성적은 '**수**' 입니다."의 형식으로 출력한다.

4. 전역변수를 사용하지 않는다.
5. 하나 이상의 함수를 사용한다.

```
<script>
    function cmdFun(kor, eng, mat) {
        var tot = kor, eng, mat;
        var avg = tot / 3;
        if(avg>=90) console.log("이번 성적은 '수' 입니다.");
        else if(avg>=80) console.log("이번 성적은 '우' 입니다.");
        else if(avg>=70) console.log("이번 성적은 '미' 입니다.");
        else if(avg>=60) console.log("이번 성적은 '양' 입니다.");
        else console.log("이번 성적은 '수' 입니다.");
    }
    cmdFun(90, 60, 50);
</script>
```

[자율 실습 예제 : 513_control-pre-ex.html]

1. prompt를 이용하여 숫자를 입력 받는다.
2. 입력을 취소한 경우 "사용자가 입력을 취소하셨습니다."를 출력한다.
3. 문자가 입력되었을 경우 "문자가 입력되었습니다."를 출력한다.
4. 입력된 숫자가 3의 배수인 경우 "3의 배수입니다"를 출력한다.
5. 입력된 숫자가 3의 배수가 아닌 경우 "3의 배수가 아닙니다."를 출력한다.
6. 아무것도 입력하지 않고 "확인"을 한 경우 "입력 값이 없습니다."를 출력한다.

[참고사항]

- null은 일반 연산의 대상은 되지 않으나 비교는 할 수 있다.
- 사용자 입력 : prompt()함수

  prompt("안내 메시지","초기값")
  자바스크립트 함수로 사용자로부터 데이터를 입력 받는다. "취소"를 선택할 경우 null을 반환한다.
  입력 없이 "확인"을 선택할 경우 빈문자열을 반환한다.
- 숫자 판별 : isNaN()

  isNaN(data)
  data가 숫자이면 false, 문자이면 true를 반환 한다.

```
<script>
    var userNum = prompt("숫자를 입력하세요.");

    if(userNum!==null) {
```

```
        if(isNaN(userNum)) {
            console.log("문자가 입력되었습니다.");
        }
        else {
            if(userNum=="") {
                console.log("입력 값이 없습니다.");
            }
            else if(userNum%3 == 0) {
                console.log("3 의 배수입니다.");
            }
            else {
                console.log("3 의 배수가 아닙니다.");
            }
        }
    }
    else {
        console.log("숫자 입력이 취소되었습니다.");
    }
</script>
```

2) switch()...case 문

switch() ... case문은 if()문과는 달리 "분기문"이라고 한다. 주어진 변수에 대응되는 case로 분기하여 블록블럭 실행하게 되어 대응되는 조건이 많은 경우 if()문 보다 우수하다.

```
switch(변수) { //일반적으로 변수가 사용되나 값을 갖고 있는 것이면 통용됨.
    case data1: 블럭문1
                break;
    case data2: 블럭문2
                break;
    ..............
    default : 블록문n
}
```

[실습예제 : 521_switch.html]

```
<script>
    var num=4;
    switch(num) {
```

```
        case 1 : console.log("입력한 숫자는 일 입니다.");
            break;
        case 2 : console.log("입력한 숫자는 이 입니다.");
            break;
        case 3 : console.log("입력한 숫자는 삼 입니다.");
            break;
        default : console.log("입력한 값이 무엇인지 모르겠습니다.");
    }
</script>
```

[응용예제 : 521_switch-ex.html]

1. 남녀의 성별을 지정한다.
2. 출력형식은 아래와 같다.
   - 2000년 이전에 태어난 남자입니다.
   - 2000년 이전에 태어난 여자입니다.
   - 2000년 이후에 태어난 남자입니다.
   - 2000년 이후에 태어난 여자입니다.
   - 내국인이 아닙니다.

```
<script>
    var jumin=4;
    switch(jumin) {
        case 1 : console.log("2000 년 이전에 태어난 남자입니다.");
            break;
        case 2 : console.log("2000 년 이전에 태어난 여자입니다.");
            break;
        case 3 : console.log("2000 년 이후에 태어난 남자입니다.");
            break;
        case 4 : console.log("2000 년 이후에 태어난 여자입니다.");
            break;
        default : console.log("내국인이 아닙니다.");
    }
</script>
```

3) for()문 : 반복문

```
for(변수=초기값 ; 반복조건 ; 변수의 증감값) {
    블록문
```

```
}
```

```
for(변수=초기값 ; 반복조건 ; 변수의 증감값) {
    for(변수=초기값 ; 반복조건 ; 변수의 증감값) {
        블록문
    }
}
```

[실습예제 : 531_for.html]

조건 1. 1부터 10까지의 합을 구하시오.
조건 2. 그 결과값을 출력하시오

```
<script>
    for(i=1;i<=10;i++) {
        tot+=i;
    }
    console.log(tot);
</script>
```

[응용예제 : 531_for-1.html]

조건 1. 1부터 10까지 출력하기.
조건 2. 출력은 아래와 같이 출력

1	2	3	4	5	6	7	8	9	10

```
<script>
    var tot=0;
    for(i=1;i<=10;i++) {
        document.write("<span>"+(tot+=i)+"</span>");
    }
</script>
```

[응용예제 : 531_for-2.html]

조건 1. 1부터 10까지의 합을 구하시오.
조건 2. 출력은 아래와 같이 출력

1	3	6	10	15	21	28	36	45	55

```
<script>
    var tot=0;
    for(i=1;i<=10;i++) {
        if(i%5==1) document.write("<p>");
        document.write("<span>"+(tot+=i)+"</span>");
        if(i%5==0) document.write("</p>");
    }
</script>
```

[응용예제 : 531_for-3.html]

조건 1. 1부터 10까지의 합을 구하시오.
조건 2. 출력은 아래와 같이 출력

1	3	6	10	15
21	28	36	45	55

```
<script>
    var tot=0;
    for(i=1;i<=10;i++) {
        if(i%5==1) document.write("<p>");
        document.write("<span>"+(tot+=i)+"</span>");
        if(i%5==0) document.write("</p>");
    }
</script>
```

[응용예제 : 531_for-4.html]

5단
5 = 5 X 1
10 = 5 X 2
15 = 5 X 3
20 = 5 X 4
25 = 5 X 5
30 = 5 X 6
35 = 5 X 7
40 = 5 X 8
45 = 5 X 9

조건 1. prompt()함수를 통해 2부터 9사이의 정수값을 입력받는다.
조건 2. 입력 받은 "단"으로 하여 구구단을 출력한다.
조건 3. 출력은 아래와 같이 출력

```
<script>
    var dan=prompt("출력할 구구단을 입력하세요");
    document.write("<p class='tit'>"+dan+"단</p>");
    for(i=1;i<10;i++) {
        document.write("<p>"+(dan*i)+" = "+dan+" X "+i+"</p>");
```

```
    }
  </script>
```

[실습예제 : 531_for-5.html]

조건 1. 구구단을 출력하세요.
조건 2. 출력은 아래와 같이 출력

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

```
<script>
    document.write("<table>");
    for(i=2;i<10;i++) {
        document.write("<tr>");
        document.write("<th>"+i+"단</th>");
        for(j=1;j<10;j++)  document.write("<td>"+i+" X "+j+" = "+(i*j)+"</td>");
        document.write("</tr>");
    }
    document.write("</table>");
</script>
```

4) while()문 / do ... while()문

① while()문
while 문은 특정 조건을 만족할 때까지 계속해서 주어진 블럭문을 반복 실행합니다. 다만 조건이 처음부터 거짓(false)이 경우에는 중괄호({ }) 안쪽의 블록문을 실행하지 않습니다.

```
while (조건) { //조건은 설명에 따라서 표현식이라고 칭하기도 함.
    블럭문;
}
```

② do ... while*()문
while 문은 루프에 진입하기 전에 먼저 조건부터 검사합니다. 하지만 do / while 문은 먼저 루프를 한 번 실행한 후에 조건을 검사합니다. 즉, do / while 문은 조건의 결과와 상관없이 무조건 한 번은 루프를 실행합니

다.

```
do {.
    블럭문;
} while (조건) ;{ //조건은 설명에 따라서 표현식이라고 칭하기도 함
```

[종합실습예제 : 541_while.html]

```
<script>
    //============= while()문으로 구구단 구현 ================================
    var i=1; //위 바깥쪽 for()문의 초기값
    var j; //안쪽 for()문에 사용 할 변수 설정
    document.write("<table><caption>구구단</caption>");
    while(i<=9) {
        document.write("<tr>");
        j=2; //아래 while()문에 사용될 변수의 초기화
        do {
            if(i==1) document.write("<th>"+j+"단</th>");
            else document.write("<td>"+j+" X "+i+" = "+(i*j)+"</td>");
            j++; //조건을 거짓으로 만들기 위해 조건에 사용된 변수값 증가
        } while(j<10) ;
        document.write("</tr>");
        i++; //조건을 거짓으로 만들기 위해 조건에 사용된 변수값 증가
    }
    document.write("</table>");
    //================== e:for()문으로 구구단 구현 ============================
</script>
```

5) 배열 반복문 : forEach()문

forEach() 반복문은 오직 Array 객체에서만 사용 가능한 메서드입니다. 배열의 요소들을 반복하여 작업을 수행할 수 있습니다. forEach 구문의 인자로 callback 함수를 등록할 수 있고, 배열의 각 요소들이 반복될 때 이 callback 함수가 호출됩니다. callback 함수에서 배열 요소의 인덱스와 값에 접근할 수 있습니다.

```
var arr = ['item1', 'item2', 'item3'];
//item은 값, index는 인덱스번호, array는 배열자체, 형식으로 반환
arr.forEach(function(item, index, array) {
    블록문;
});
```

```
arr.forEach((item, index, array) => 함수()); //위와 동일한 형식
```

```
<script>
	var arr= ['apple', 'banana', 'kiwi', 'melon'];
	arr.forEach(function (val, idx, array) {//값, 인덱스, 배열 을 반환 여기서 배열은 의미 없음.
		console.log(val, idx, array);
	});
	//아래 화살표 형식으로 변환 가능
	arr.forEach((val, idx) => console.log(val, idx)); //array 는 추출하지 않음
</script>
```

6) 배열 반복문 : for / in

배열의 개수만큼 반복하며 인덱스 반환

```
for (변수 in 객체) {
    블럭문;
}
```

```
<script>
	var arr= ['apple', 'banana', 'kiwi', 'melon'];
	for (var inx in arr) {
		console.log(idx); //인덱스 반환
	}
</script>
```

7) 객체 반복문 : for / of

배열의 개수만큼 반복하며 인덱스 반환

```
for (변수 of 객체) {
    블럭문;
}
```

```
<script>
	var arr= ['apple', 'banana', 'kiwi', 'melon'];
	for (var val of arr) {
		console.log(val); //각 방의 데이터 반환
	}
```

```
</script>
```

6. 배열

배열(array)은 이름과 인덱스로 참조되는 정렬된 값의 집합으로 정의됩니다. 배열을 구성하는 각각의 값을 배열 요소(element)라고 하며, 배열에서의 위치를 가리키는 숫자를 인덱스(index)라고 합니다.

[배열의 특징]
1. 배열 요소의 타입이 고정되어 있지 않으므로, 같은 배열에 있는 배열 요소간의 타입이 서로 다를 수도 있습니다.
2. 배열 요소의 인덱스가 연속적이지 않아도 되며, 따라서 특정 배열 요소가 비어 있을 수도 있습니다.
3. 자바스크립트에서 배열은 Array 객체로 다뤄집니다.

1) 배열의 선언

```
var arr = [ data1, data2, data3, ... ]; //배열 리터럴로 배열을 선언하고 각 저장공간에 초기값 대입
var arr = new Array(n); //n은 저장공간의 갯수
var arr = new Array(data1, data2, data3, ...); //배열명을 선언하고 각 저장공간에 초기값 대입
```

2) 배열의 특성

자바스크립트의 배열은 배열명을 출력 대상으로 할 경우 각 방에 있는 데이터를 콤마(,)로 구분하여 반환을 합니다.

3) 배열의 주요 속성 및 메서드

① 속성

• Array.length : 배열의 길이(방의 개수)를 반환

② 배열 메서드

• Array.isArray() : 메소드는 전달받은 값이 Array 객체인지 아닌지를 검사합니다.

```
Array.isArray(배열명)
```

• push() : 배열의 마지막부터 값을 추가합니다.

```
var arr = ['apple', 'banana', 'kiwi', 'melon'];
arr.push('pear', 'berry'); //['apple', 'banana', 'kiwi', 'melon', 'pear', 'berry']가 됩니다.;
```

• pop() : 배열의 가장 마지막 값을 삭제합니다.

```
var arr = ['apple', 'banana', 'kiwi', 'melon'];
arr.pop(); //['apple', 'banana', 'kiwi']가 됩니다.;
```

• unshift() : 배열의 가장 앞에서부터 값을 추가합니다. (속도가 느림)

```
var arr = ['apple', 'banana', 'kiwi', 'melon'];
arr. unshift ('pear', 'berry'); //['pear', 'berry', 'apple', 'banana', 'kiwi']가 됩니다.;
```

• shift() : 배열의 가장 앞에 값을 삭제합니다. (속도가 느림)

```
var arr = ['apple', 'banana', 'kiwi', 'melon'];
arr. unshift (); //['banana', 'kiwi']가 됩니다.;
```

• splice() : 배열에서 원하는 인덱스의 값을 삭제하거나 삭제하고 추가할 수 있습니다.

```
//형식 : ArrayName.splice(시작인덱스, 삭제갯수,data-1, data-1, .....);
var arr = ['pear', 'berry', 'apple', 'banana', 'kiwi'];
arr.splice(1,2); //['pear', 'banana', 'kiwi']가 됩니다.;
arr.splice(1,1, 'kiwi', 'banana', 'berry'); //['pear', kiwi', banana', berry', 'kiwi']
arr.splice(); ////['pear', kiwi', banana']
```

• concat() : 두 개의 배열을 연결합니다.(순서대로 병합)

```
//형식 : ArrayName.concat(배열명);
var arr1 = ['pear', 'berry', 'apple', 'banana', 'kiwi'];
var arr2 = [9, 7, 5, 3];
var merArr = arr1.concat(arr2); // ['pear', 'berry', 'apple', 'banana', 'kiwi', 9, 7, 5, 3]
```

• indexOf() : 배열의 처음부터 검색하여 동일한 값의 인덱스를 반환, 발견하지 못할 경우 -1 반환

```
//형식 : ArrayName.indexOf(data);
var arr = ['pear', 'berry', 'banana', 'banana', 'kiwi'];
var arrIdx = arr.indexOf("banana"); // 처음 banana가 발견된 방의 인덱스 반환 : 2
```

• lastIndexOf() : 배열의 마지막부터 검색하여 동일한 값의 인덱스를 반환, 발견하지 못할 경우 -1 반환

```
//형식 : ArrayName.indexOf(data);
var arr = ['pear', 'berry', 'banana', 'banana', 'kiwi'];
var arrIdx = arr. lastIndexOf ("banana"); // 처음 banana가 발견된 방의 인덱스 반환 : 3
```

• includes() : 데이터가 배열에 포함되어 있으면 true, 없으면 false 반환

```
//형식 : ArrayName.indexOf(data);
var arr = ['pear', 'berry', 'banana', 'banana', 'kiwi'];
var arrChk = arr. includes ("banana"); // banana가 있으므로 true 반환
var arrChk2 = arr. includes ("waterMelon"); // waterMelon가 없으므로 false 반환
```

7. String 객체

자바스크립트에서 문자열 리터럴은 큰따옴표("")나 작은따옴표('')를 사용하여 손쉽게 만들 수 있습니다.

1) 이스케이프 문자

문자	설명	문자	설명
\’	작은 따옴표	\r	캐리지 리턴
\”	큰 따옴표	\t	텝
\\	역슬래스	\b	역행 키이
\n	새로운 라인	\f	용지 공급

example : 711_string.html

```
var str = "₩"산토끼₩" 토끼야 어디를 가느냐?₩r₩n₩'깡총깡총₩' 뛰면서 어디를 가느냐?";
console.log(str);
```

2) 메서드

형식

```
“문자열”.메서드();
```

메소드와 사용법	역할
charAt(index)	지정된 위치에 있는 문자 반환
indexOf(string)	지정된 문자의 위치를 왼쪽부터 찾아 위치값 반환
lastIndexOf(string)	지정된 문자의 위치를 오른쪽부터 찾아 위치값 반환
substring(index1, index2)	문자열에서 지정한 위치의 문자열 추출
substr(start_index, length)	문자열에서 지정한 위치의 문자열 추출
toLowerCase()	소문자로 변환하기
toUpperCase()	대문자로 변환하기
concat(string)	두 문자열을 합치기
slice(start_index, end_index)	문자열의 일부를 추출하기

replace(searchValue, newValue)	문자열 치환(왼쪽부터 찾은 첫번째 문자열만 치환)
replaceAll(searchValue, newValue)	문자열 치환(찾은 전체 문자열을 모두 치환)
split([분리자])	문자열을 분리하여 배열로 반환
charCodeAt([index])	문자열의 ISO Latin-1 값 알아내기
fromCharCode("n1", ..., "nn")	ISO Latin-1 값의 문자열 알아내기

① 문자열에서 지정된 위치에 있는 문자 반환

- **charAt() :** 문자열에서 지정된 인덱스 위치의 문자를 반환합니다.
- **charCodeAt() :** 문자열에서 지정된 인덱스 위치의 문자를 UTF-16 코드를 반환.
- **charPointAt() :** 문자열에서 지정된 인덱스 위치의 문자를 UTF-16 코드를 반환.

```
char str.cahrAt(int index)
int str.charCodeAt(int index)
int str.charPointAt(int index)
```

- str : 문자열
- index : 0과 (문자열의 길이 - 1) 사이의 정수값, 기본값은 0으로 생략하면 첫 문자 반환
- 반환값 : 유니코드 단일문자, index가 문자열 길이보다 큰 경우 빈문자열 반환

example : 711_string_1.html

```
var str = "Hare rabbit, where are you going? Where are you going while running?";
var resutl1, resutl2, resutl3;

var result1 = str.charAt(6); // 변수 str에서 6번째 위치한 문자 반환 ==> a
var result2 = str.charCodeAt(6); // 변수 str에서 6번째 위치한 문자(a) UTF-16반환 ==> 97
var result3 = str.codePointAt(7); // 변수 str에서 7번째 위치한 문자(b) UTF-16 코드 ==> 98

console.log(result1, result2, result3); // "끼"를 콘솔에 출력
```

② 특정 문자열의 위치 반환

- **indexOf() :** 문자열에서 지정한 문자열을 왼쪽부터 찾아 위치 값을 반환
- **lastIndexOf() :** 문자열에서 지정한 문자열을 오른쪽부터 찾아 위치 값을 반환

```
int str.indexOf(string)
int str.lastIndexOf(string)
```

- str: 문자열

- string: 찾는 문자열
- 반환 값: 첫 번째 찾은 문자열의 index를 반환하고, 찾지 못한 경우 -1을 반환함.

example : 712_string_2.html

var str = "산토끼 토끼야 어디를 가느냐? 깡총깡총 뛰면서 어디를 가느냐";
var idx1 = str.indexOf("토끼"); ➔ 외쪽부터 "토끼"를 찾아 index를 반환 ➔ **1 반환**
var idx2 = str.lastIndexOf("토끼"); ➔ 오른쪽 부터 "토끼"를 찾아 index를 반환 ➔ **4 반환**

console.log(idx1, idx2); ➔ idx를 콘솔에 출력

③ 문자열 추출

- **slice() :** 문자열에서 지정된 위치의 문자열을 지정된 위치만큼 선택하여 반환
- **substring() :** 문자열에서 지정된 위치의 문자열을 지정된 위치만큼 선택하여 반환
- **substr() :** 문자열에서 지정된 위치의 문자열을 지정된 개수 만큼 선택하여 반환

String str.slice(int index1[, intindex2])
String str.substring(int index1[, int index2])
String str.substr(int index[, int length])

- str: 문자열
- index 1 : 반환 할 문자열의 시작위치
- index 2 : 반환 할 문자열의 마지막 위치(마지막 문자열의 위치는 "index 2 - 1"의 위치)
  index 1를 생략하면 index 1번 문자열부터 끝 문자열까지 반환
- length : 문자열의 길이(개수)
- 반환 값: String 또는 빈 문자열

example : 713_string_3.html

var str = "산토끼 토끼야 어디를 가느냐? 깡총깡총 뛰면서 어디를 가느냐";
var result1 = str.slice(-16, -12); ➔ slice()는 음수가 지정하면 오른쪽에서부터 위치를 나타낸다.
-16번째 문자부터 -12번째 앞 문자까지 반환
var result2 = str.substring(17, 21); ➔ str의 17번째 문자인 "깡"부터 21번째 이전 글자인 "총"까지 반환
var result3 = str.substr(17, 4); ➔ str의 17번째 문자인 "깡"부터 4개 글자 반환
console.log(result1, result2, result3); ➔ 깡총깡총, 깡총깡총, 깡총깡총

④ 대소문자 변환

- **toLowerCase():** 문자열을 모두 소문자로 변환
- **toUpperCase():** 문자열을 모두 대분자로 변환

String str.toLowerCase()
String str.toUpperCase()

- str: 문자열
- 반환 값: String 또는 빈 문자열

example : 714_string_4.html

```
var str = "Hare rabbit, where are you going? Where are you going while running?";
var result1 = str.toLowerCase(); ➔ str의 모든 영문자를 소문자로 변환
var result2 = str.toUpperCase(); ➔ str의 모든 영문자를 대문자로 변환
console.log(result1);
console.log(result2);
```

⑤ 문자열 바꾸기

- **replace() :** 문자열 중 처음 찾은 문자열을 지정한 문자열로 바꿈.
- **replaceAll() :** 문자열에서 찾은 문자열을 지정한 문자열로 모두 바꿈

String str.replace(string str1, string str2)
String str.replaceAll(string str1, string str2)

- str: 문자열
- str1: 찾는 문자열
- str2 : 바꿀 문자열
- 반환 값: String

example : 715_string_5.html

```
var str = " 산토끼 토끼야 어디를 가느냐? 깡총깡총 뛰면서 어디를 가느냐?";
var result1 = str.replace("토끼", "거북이"); ➔ str의 첫 "토끼"를 "거북이"로 바꿈
var result2 = str.replaceAll("토끼", "거북이"); ➔ str의 모든 "토끼"를 "거북이"로 변환
console.log(result1);
console.log(result2);
```

⑥ 문자열 나누기(배열화)

- **split() :** 지정된 문자열을 분리문자를 기준으로 분리 및 배열화 함

Array str.split([string]) ➔ String arr.join([string])

- str: 문자열
- string: 분리 문자열

- 반환 값: Array
- 참고 : Array.join() 메서드는 split()메서드와 반대의 역할을 수행함.

example : 716_string_6.html

```
var str = "메론, 수박, 자두, 포도, 자몽, 사과, 복숭아";
var arr = str.split(", "); ➔ str의 문자열을 ", "를 분리 문자열로 배열화 하여 Array로 반환
console.log(arr);
```

⑥ 문자열 연결

• **concat() :** 문자열을 순서대로 연결함.

String str.concat([string])

- str: 문자열
- string: 뒤쪽에 연결할 문자열
- 반환 값: String

example : 717_string_7.html

```
var strKor = " 산토끼 토끼야 어디를 가느냐? 깡총깡총 뛰면서 어디를 가느냐?";
var strEng = "Hare rabbit, where are you going? Where are you going while running?";
var result = strKor.concat("₩r₩n");
result = strKor.concat("("+strEng+")");
console.log(arr);
```

3) 속성

문자열의 길이를 나타낸다. 반환만 가능하며 수정 할 수 없음.

• **length :** 문자열 길이 반환.

int str.length

- str: 문자열
- 반환 값: int

example : 731_string.html

```
var str = " 산토끼 토끼야 어디를 가느냐? 깡총깡총 뛰면서 어디를 가느냐?";
var result = str.length;
```

```
console.log(result);
```

8. Date 객체

자바스크립트에서는 Date 객체를 사용하여 매 순간 변화하는 시간과 날짜에 관한 정보를 손쉽게 얻을 수 있습니다.
Date 객체는 연월일, 시분초의 정보와 함께 밀리초(millisecond)의 정보도 함께 제공합니다.

1) Date 객체의 생성

형식

- new Date()
- new Date("날짜를 나타내는 문자열")
- new Date(밀리초)
- new Date(년, 월, 일, 시, 분, 초, 밀리초)

example : 810_date.html

```
var now = new Date(); //현재의 날자 및 시간을 반환
var choDate1 = new Date("2022-09-10 00:00:00");
//new Date(year, month, day, hours, minutes, seconds, milliseconds)
var choDate2 = new Date(2022, 9, 11, 13, 30, 10, 1000);

console.log(now);
console.log(choDate1);
console.log(choDate2);
```

2) 문자열로 변환하여 출력하는 기본함수

메소드와 사용법	역할
`toString()`	Sat Sep 10 2022 22:02:26 GMT+0900 (한국 표준시)
`toDateString()`	Sat Sep 10 2022
`toLocaleString()`	2022. 9. 10. 오후 10:02:26
`toLocaleDateString()`	2022. 9. 10.

example : 820_date.html

```
<script>
	var now = new Date(); //현재의 날자 및 시간을 반환
```

```
    console.log(now.toString());
    console.log(now.toDateString());
    console.log(now.toLocaleString());
    console.log(now.toLocaleDateString());
</script>
```

3) 날짜 가져오기

메소드와 사용법	역할
int getFullYear()	현지 시각으로 현재 연도를 4 자리의 숫자(YYYY)로 반환함.
int getYear()	현재 시각으로 현재 연도를 3 자리 숫자(YYY)로 반환함
int getMonth()	현지 시각으로 현재 월에 해당하는 숫자를 반환함.(월 - 1)로 반환함
int getDate()	현지 시각으로 현재 일자에 해당하는 숫자를 반환함.
int getDay()	현지 시각으로 현재 요일에 해당하는 숫자를 반환함.
int getHours()	현지 시각으로 현재 시각에 해당하는 숫자를 반환함.
int getMinutes()	현지 시각으로 현재 시각의 분에 해당하는 숫자를 반환함.
int getSeconds()	현지 시각으로 현재 시각의 초에 해당하는 숫자를 반환함.
int getMilliseconds()	현지 시각으로 현재 시각의 밀리초에 해당하는 숫자를 반환함.
int getTime()	1970 년 1 월 1 일 0 시 0 분 0 초부터 현재까지의 시간을 밀리초 단위로 환산한 값을 숫자로 반환함.
int getTimezoneOffset()	UTC 로부터 현재 시각까지의 시간차를 분 단위로 환산한 값을 숫자로 반환함.

example : 821_date.html

```
var nowDate = new Date(); //현재의 날자 및 시간을 반환
var nowYear = nowDate.getFullYear(); //YYYY로 반환
var nowMonth = nowDate.getMonth()+1; //월-1 반환
var nowDay= nowDate.getDate(); //일자
var nowWeek = nowDate.getDay(); //일요일-0, 토요일-6
var longWeek, shotWeek; //요일을 저장할 변수 지정
```

```
switch(nowWeek) {
      case 0 : longWeek="일요일"; showWeek="일"; break;
      case 1 : longWeek="월요일"; showWeek="월"; break;
      case 2 : longWeek="화요일"; showWeek="화"; break;
      case 3 : longWeek="수요일"; showWeek="수"; break;
      case 4 : longWeek="목요일"; showWeek="목"; break;
      case 5 : longWeek="금요일"; showWeek="금"; break;
      default : longWeek="토요일"; showWeek="토"; break;
}
var strDate = nowYear+ "년 " + nowMonth + "월 " + nowDay + "일 " + longWeek;

console.log(strDate);
```

example : 822_time.html

```
<script>
      var nowDate = new Date(); //현재의 날자 및 시간을 반환
      var nowYear = nowDate.getFullYear(); //YYYY로 반환
      var nowMonth = nowDate.getMonth()+1; //월-1 반환
      var nowDay= nowDate.getDate(); //일자
      var nowWeek = nowDate.getDay(); //일요일-0, 토요일-6

      var nowHour = nowDate.getHours();
      var nowMinute = nowDate.getMinutes();
      var nowSecond = nowDate.getSeconds();

      var strDate = nowYear+ "년 " + nowMonth + "월 " + nowDay + "일 ";
      var strTime = nowHour + ":" + nowMinute + ":" nowSecond;

      console.log(strDate, strTime);
</script>
```

4)날짜 설정하기

메소드와 사용법	역할
int setFullYear()	연도를 4 자리의 숫자(YYYY)로 설정함.
int setMonth()	월에 해당하는 숫자를 설정함.(월 - 1)로 설정

int setDate()	일자에 해당하는 숫자를 설정함.
int setHours()	시각을 설정함.
int setMinutes()	분을 설정함.
int setSeconds()	초를 설정함.
int setMilliseconds()	밀리초를 설정함.

example : 841_date_set.html

```
<script>
    var nowDate = new Date(); //현재의 날자 및 시간을 반환

    nowDate.setFullYear(2021);
    nowDate.setMonth(9);
    nowDate.setDate(5);

    nowDate.setHours(13);
    nowDate.setMinutes(10);
    nowDate.setSeconds(30);
    nowDate.setMilliseconds(2000);

    console.log(nowDate.toString());
</script>
```

9. Math 객체

Math 객체는 수학에서 자주 사용하는 상수와 함수들을 미리 구현해 놓은 자바스크립트 표준 내장 객체입니다.

Math 객체는 다른 전역 객체와는 달리 생성자(constructor)가 존재하지 않습니다. 따라서 따로 인스턴스를 생성하지 않아도 Math 객체의 모든 메소드나 프로퍼티를 바로 사용할 수 있습니다.

1) 주요 메서드

주요 메서드	역할
Math.min()	최소값 구하기
Math.max()	최대값 구하기
Math.random()	0 ~ 1 사이의 난수 구하기
Math.round()	반올림
Math.floor()	내림
Math.ceil()	올림
Math.sin()	삼각 함수 중 sin 값 구하기

① 최대값 최소값 구하기

Number Math.max(n1, n2, ...) | Number Math.min(n1, n2, ...)

- Number : 숫자
- n1, n2, ... : 숫자
- return : Number

example : 911_math.html

```
<script>
    var minNum = Math.min(2, 3, 8.965, 100, 7, -3);
    var maxNum = Math.max(2, 3, 8.965, 100, 7, -3);

    console.log("Max : "+maxNum, "Min : "+minNum)
</script>
```

② 소수점 올림, 반올림, 내림

int Math.ceil(num) | int Math.round(num) | int Math.floor(num)

- num : 숫자
- return : int

example : 912_math.html

```
<script>
    var num1 = Math.ceil(4.1);
    var num2 = Math.ceil(-4.1);
    var num3 = Math.round(4.1);
    var num4 = Math.round(4.5);
    var num5 = Math.round(-4.1);
    var num6 = Math.round(-4.5);
    var num7 = Math.floor(4.9);
    var num8 = Math.floor(-4.9);

    console.log("ceil(4.1):"+num1, "ceil(-4.1) : "+num2);
    console.log("round(4.1):"+num3, "round(4.5) : "+num4, "round(-4.1) : "+num5, "round(-4.5) : "+num6);
    console.log("floor(4.9):"+num7, "floor(-4.9) : "+num8);
</script>
```

③ 난수 구하기 : 0이상 1미만의 수

Number Math.random()

- return : Number

example : 913_math.html

```
<script>
    var num = Math.random();
    console.log(num);
</script>
```

example : 913_exam_random.html

```
<script>
    var num = Math.random()*101;
    num = Math.floor(num);
    console.log(num);
```

```
</script>
```

2) 기타 메서드

메서드	역할
Math.abs(x)	x 의 절댓값을 반환함.
Math.cbrt(x)	x 의 세제곱근을 반환함.
Math.sqrt(x)	x 의 제곱근을 반환함.
Math.clz32(x)	x 을 32 비트 이진수로 변환한 후, 0 이 아닌 비트의 개수를 반환함.
Math.exp(x)	e^x 의 값을 반환함. (e : 오일러의 수)
Math.expm1(x)	1 - e^x 의 값을 반환함.
Math.fround(x)	x 와 가장 근접한 32 비트 부동 소수점 수(single precision float)를 반환함.
Math.log(x)	x 의 자연로그 값을 반환함. (ln x)
Math.trunc(x)	x 의 모든 소수 부분을 삭제하고 정수 부분만을 반환함.
Math.pow(x, y)	x 의 y 승을 반환함.
Math.sin(x), Math.cos(x), Math.tan(x),Math.asin(x), Math.acos(x), Math.atan(x), Math.asinh(x), Math.acosh(x), Math.atanh(x), Math.atan2(x)	삼각함수

3) Math 프로퍼티

자바스크립트는 수학에서 사용하는 다양한 상수들을 Math 프로퍼티를 이용해 제공하고 있습니다.

프로퍼티	설명	대략값
Math.E	오일러의 수(Euler's constant)라고 불리며, 자연로그(natural logarithms)의 밑(base) 값	2.718

프로퍼티	설명	대략값
Math.LN2	2 의 자연로그 값	0.693
Math.LN10	10 의 자연로그 값	2.303
Math.LOG2E	오일러 수(e)의 밑 값이 2 인 로그 값	1.443
Math.LOG10E	오일러 수(e)의 밑 값이 10 인 로그 값	0.434
Math.PI	원의 원주를 지름으로 나눈 비율(원주율) 값	3.14159
Math.SQRT1_2	2 의 제곱근의 역수 값	0.707
Math.SQRT2	2 의 제곱근 값	1.414

10. Number 객체

자바스크립트에서는 정수와 실수를 따로 구분하지 않고, 모든 수를 실수 하나로만 표현합니다.

1) 진법의 표현

자바스크립트에서는 기본적으로 10진법을 사용하여 수를 표현합니다. 하지만 0x 접두사를 사용하여 16진법으로 수를 표현할 수도 있습니다.

형식

- 2진수(**0b**xx) | 8진수(**0o**xx) | 10진수(숫자) | 16진수(**0x**xx)

example : 1010_number.html

```
<script>
        var aa = 0b11;    // 2진법(binary) # (1*2) + (1*1) = 3
        var bb = 0o23;   // 8진법(octal)  # (2*8) + (3*1) = 19
        var cc = 34;       // 10진법(decimal) # (3*10) + (4*1) = 34
        var dd = 0x45;   // 16진법(hexadecimal) # (4*16) + (5*1) = 69
         // 두 수 모두 10진법으로 자동으로 변환되어 계산됨. -> 125
        console.log(aa+"+"+bb+"+"+cc+"+"+dd+"="+(aa + bb +  cc + dd));
    </script>
```

2) toString()를 통한 진수의 변환

toString() 메소드를 사용하여 해당 숫자를 여러 진법의 형태로 변환할 수 있습니다.

String num.toString(digit)

- num : 수자 변수
- digit : 표현할 진수 선택
- return : 지정된 진수에 따른 num을 변환하여 String으로 반환

example : 1020_number.html

```
<script>
    var num = 255;
    console.log("10 진수 "+ num +"은 2 진수 "+num.toString(2)+"이다!!!");
    console.log("10 진수 "+ num +"은 8 진수 "+num.toString(8)+"이다!!!");
    console.log("10 진수 "+ num +"은 10 진수 "+num.toString(10)+"이다!!!");
```

```
    console.log("10 진수 "+ num +"은 16 진수 "+num.toString(16)+"이다!!!");
</script>
```

3) null, undefined, NaN, Infinity

값	Boolean 문맥	Number 문맥	String 문맥
null	false	0	"null"
undefined	false	NaN	"undefined"
NaN	false	NaN	"NaN"
Infinity	true	Infinity	"Infinity"

example : 1030_number.html

```
<script>
    typeof null;        // object
    typeof undefined;   // undefined
    typeof NaN;         // number
    typeof Infinity;    // number

    Boolean(null);      // false
    Boolean(undefined); // false
    Boolean(NaN);       // false
    Boolean(Infinity);  // true

    Number(null);       // 0
    Number(undefined);  // NaN
    Number(NaN);        // NaN
    Number(Infinity);   // Infinity

    String(null);       // null
```

```
    String(undefined);  // undefined
    String(NaN);        // NaN
    String(Infinity);   // Infinity
</script>
```

4) 주요 Number Method

메소드	설명
Number.parseFloat()	문자열을 파싱하여, 문자열에 포함된 숫자 부분을 실수 형태로 반환함.
Number.parseInt()	문자열을 파싱하여, 문자열에 포함된 숫자 부분을 정수 형태로 반환함.
Number.isNaN()	전달된 값이 NaN 인지 아닌지를 검사함.
Number.isFinite()	전달된 값이 유한한 수인지 아닌지를 검사함.
Number.isInteger()	전달된 값이 정수인지 아닌지를 검사함.
Number.isSafeInteger()	전달된 값이 안전한 정수(safe integer)인지 아닌지를 검사함.

위 Method는 모두 전역함수를 지원하며, 실제 전역함수를 사용할 것을 권장 합니다.

example : 1040_number.html

```
<script>
    var Num1 = Number.parseFloat("12.34");      // 12.34
    var Num2 = Number.parseInt(12.34); //12
    var Num3 = Number.isNaN("abc"); //false
    var Num5 = Number.isFinite(undefined); //false
    var Num6 = Number.isInteger("12.34"); //false , 정수인 경우에만 true
    var Num7 = Number.isSafeInteger("12.34"); //false

    // 실제 개발에 있어 전역변수의 사용이 더 많이 쓰임
    // 데이터를 강제적으로 숫자로 변경하여 사용하게 됨.
```

```
    var num1 = parseFloat("12.34");      // 12.34
    var num2 = parseInt(12.34); //12
    var num3 = isNaN("abc"); //true
    var num5 = isFinite(undefined); //false
    var num6 = isInteger("12.34"); //false , 정수인 경우에만 true
    var num7 = isSafeInteger("12.34"); //false

    console.log(Num1, Num2, Num3, Num4, Num5, Num6, Num7);
    console.log(num1, num2, num3, num4, num5, num6, num7);
</script>
```