

# day19-JavaScript

☰ 태그	
📅 날짜	@2022년 10월 26일

## JavaScript

브랜든 아이크가 1995년 10일만에 설계를 마쳤다는 것으로 시작했다. Mocha 라는 이름으로 시작하여 LiveScript 3달 뒤에 JavaScript으로 이름이 정해졌다.

자바스크립트는 자바와는 전혀 별개의 언어이다.

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    console.log("Hello World... js"); //세미콜론 생략가능하지만 언제나 넣을것
  </script>
</body>
</html>
```

자바 스크립트 소스 코드는 html 문서 내의 <script> 태그로 기술한다.

<h1> 처럼 자바 스크립트 소스 코드 내에서도 html태그를 기술할 수 있다.

document.write() 명령문을 통해 화면에 출력한다.

```
<script>
  document.write("<h1>Hello World... js hello</h1>"); //세미콜론 생략가능하지만 언제나 넣을것
</script>
```

## JS의 위치

1. <head>에 위치
2. <body>에 위치
3. 외부 파일에 기술

JS.JS

```
document.write("외부 파일입니다.<br>");  
document.write("외부 파일의 확장자는 js로 설정합니다.");
```

```
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta http-equiv="X-UA-Compatible" content="IE=edge">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Document</title>  
</head>  
<body>  
  <script src="JS.JS"></script>  
</body>  
</html>
```

세미콜론 ;

자바 스크립트에서는 세미콜론이 선택사항이지만 반드시 사용하자.

소스 코드는 한 줄에 하나의 구문을 사용하는 것이 좋다.

들여쓰기(만입)

자바 스크립트도 자바와 같이 들여쓰기를 문제 삼지는 않는다. 특별한 규약은 없다.

들여쓰기를 체계화하여 사용하면 가독성이 좋아지기 때문에 사용을 강력 권장

4칸 들여쓰기를 권장

(tab사용은 대부분의 커뮤니티에서는 권장하지 않는다.)

주석

코드에 대한 설명 또는 잠시 코드 실행을 멈추고 버그나 오타등을 확인 하는 용도로 많이 사용된다.

자바와 명령어는 같다.

`/* */` : 여러줄 주석

`//` : 한줄 주석

```
주석 실습
TEST.html

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    /*
      코드 작성일: 5월 25일
      코드 작성자 : 이병재
      코드 관리자 : 서혜숙
    */
    document.write("소스 코드는 1개 라인에 1개의 문을 기술한다.<br>"); /*첫번째 출력 */
    document.write("문의 제일 뒤에는 세미콜론을 기술한다<br>"); // 두번째 출력
    document.write("문은 공식적인 용어이며 명령문이라고 불려도된다."); //세번째 출력
  </script>
</body>
</html>
```

## 대문자와 소문자 구분

자바 스크립트 소스 작성 시 대소문자의 구분은 필수이다.

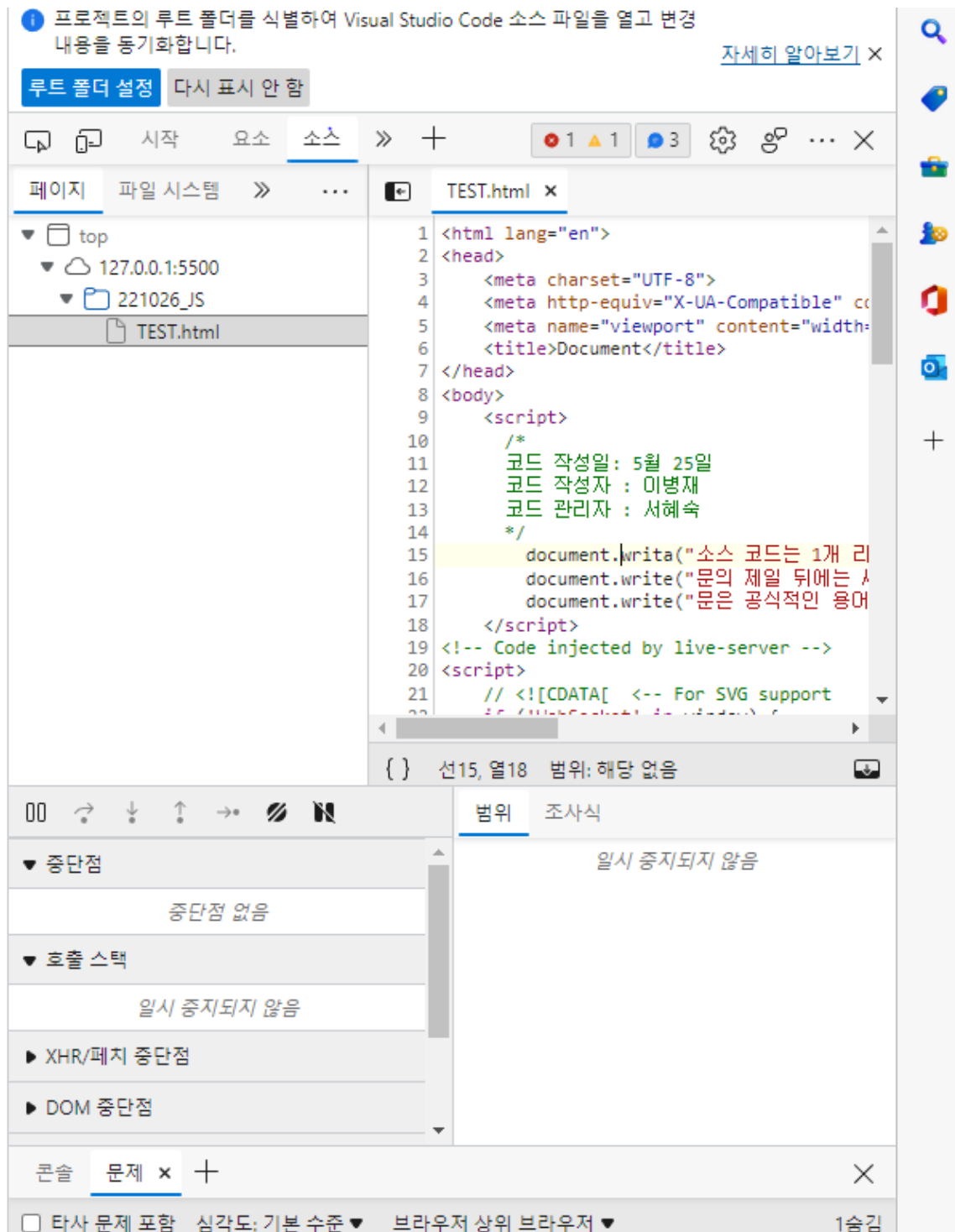
※ 오타가 날 경우 에러난 부분을 찾기가 힘들다.

찾을 수 있는 방법이 있긴하다.

버그/오타 등을 수정하는 것을 ‘**디버그**’라고 한다.

F12키를 눌러서 에러가 난 지점을 확인 가능하다.

아래 사진 처럼 에러가 난 지점을 노란색으로 표시해준다.



## 변수와 상수

TEST1.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>

    var x, y, z;    // x, y, z 세개의 변수를 선언
    x = 100;        // 숫자 100을 x에 할당
    y = 90;         // 숫자 90을 y에 할당
    z = x + y;      // 변수 x와 y의 값을 더한 결과

    // 내부적으로 자료형을 자동 할당 하는것이다.

    document.write(z); // z를 화면에 출력

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

## 변수 이름 규칙

1. 문자와 숫자로도 사용 가능하다.
2. 특수 문자는 \_, \$ 만 사용 가능하다. 첫글자로도 가능한데 첫 글자로는 사용하지 않는것이 좋다.
3. 숫자로는 시작 하지 않는다.
4. 대소문자를 완전히 구분한다. a = 7, A = 8 은 다른 변수가 된다.
5. 변수명은 여러 단어의 의미를 살려 사용하는 것이 좋다. 자바와는 달리  
자바) firstValue JS) first\_value 처럼 선호하는 변수의 모양이 다르다.
6. 예약어는 사용할 수 없다.

## JavaScript에서의 자료형 할당

TEST2.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>

    var test;

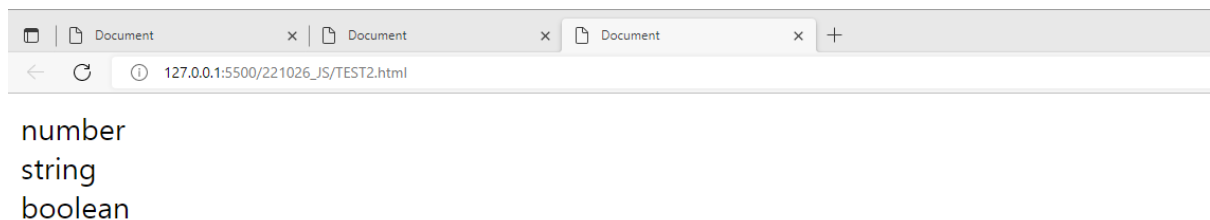
    test = 10.5;
    document.write(typeof test + "<br>");

    test = "school";
    document.write(typeof test + "<br>");

    test = true;
    document.write(typeof test);

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

// 아래 사진 처럼 자동으로 자료형을 할당해주는 것을 확인할 수 있다.



test 변수에 선언만 하고 아무런 값도 할당하지 않고 그냥 자료형과 값을 출력해 보자.  
결과는 둘 다 undefined로 출력이 되었다.

TEST2.html

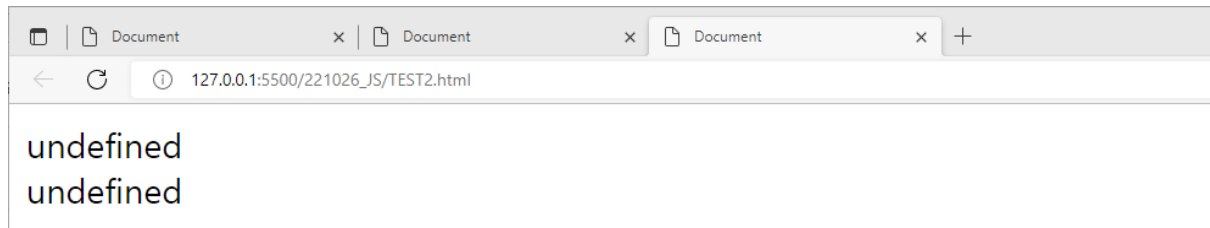
```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
```

```

</head>
<body>
  <script>
    var test;
    document.write(typeof test + "<br>");
    document.write(test);

  </script>
</body>
</html>

```



※ **동적할당**은 변수에 입력되는 값에 따라 그 변수의 자료형이 결정되기 때문에 값을 넣어주지 않으면 ‘미정의 undefined’ 상태로 출력이 된다.

## 숫자

소수점이 없는 정수와 소수점이 있는 실수

```

TEST3.html

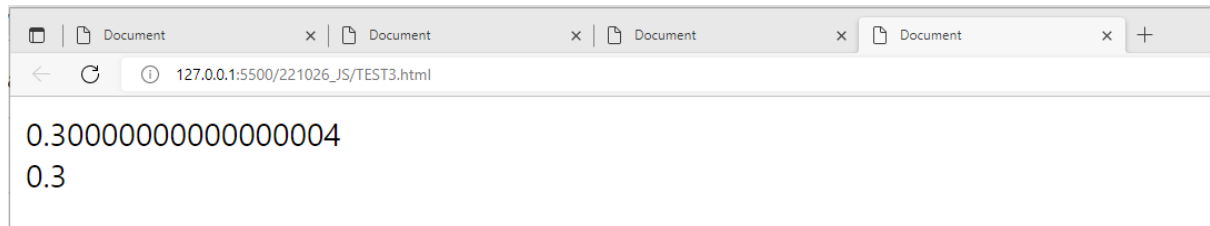
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var x = 0.2 + 0.1;
    document.write(x);

    document.write("<br>");

    var y = (0.2*10 + 0.1*10) / 10;
    document.write(y);
  </script>
</body>
</html>

```

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



결론적으로 계산식은 같다. 하지만 두개의 식이 다른 결과가 나온다.

이유는 **부동 소수점** 이다.

컴퓨터 상에서 진수를 변환하면서 **미세한 오차**가 발생한다.

해결법으로는 계산식을 **정수화 시켜서 계산후에 다시 실수화**를 시키면 된다.

## Infinity & NaN

- Infinity

0으로 나누었을 때 엄청나게 큰수가 발생했을 경우에 반환하는 숫자 상수이다.

JS에서는 문자가 아닌 숫자 number로 취급한다. 마치 문법적으로 하나의 숫자 상수처럼 사용된다.

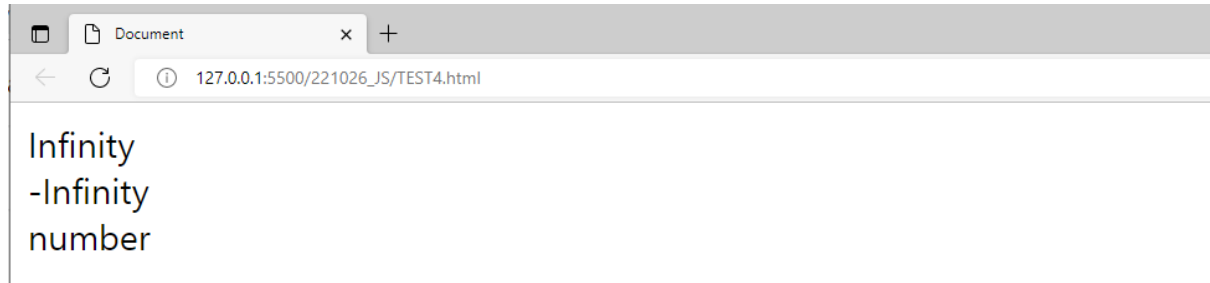
TEST4.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var x = 2/0;
    document.write(x);
    document.write("<br>");

    var y = -2/0;
    document.write(y);
    document.write("<br>");
```



```
document.write(typeof Infinity)
</script>
</body>
</html>
```



- NaN (Not a Number)

숫자가 아니라는 의미지만 문법적으로는 숫자로 취급한다.

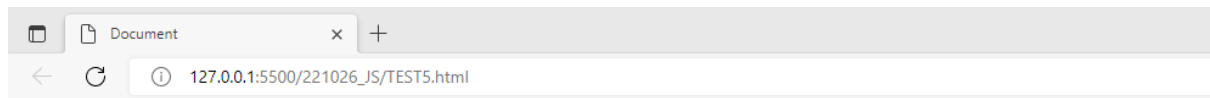
```
TEST5.html

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var x = 100;
    var y = "count";

    document.write(x-y);
    document.write("<br>");

    document.write(typeof NaN);

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



NaN  
number

- String 문자열

JS에서는 “” 쌍따옴표와 “” 홑따옴표를 똑같이 취급한다.

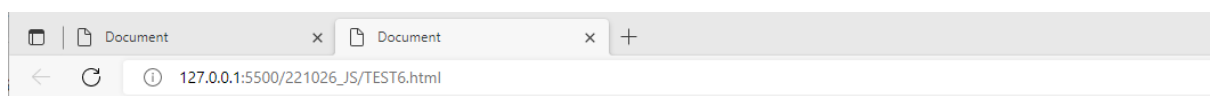
TEST6.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var car1 = "SUV";
    document.write(car1 + "<br>");

    var car2 = 'JEEP';
    document.write(car2 + "<br>");

    document.write(car1 + " " + car2);

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



SUV  
JEEP  
SUV JEEP

## 문장 내의 “”와 “” 사용

만약 문장 안에서 austin's car처럼 홑따옴표를 출력하고 싶을 때는 문장 전체를 쌍따옴표로 닫아준다.

→ "austin's car"

마찬가지로 문장 안에서 say "HI" 쌍따옴표를 출력하고 싶을 때는 문장 전체를 홑따옴표로 닫아준다.

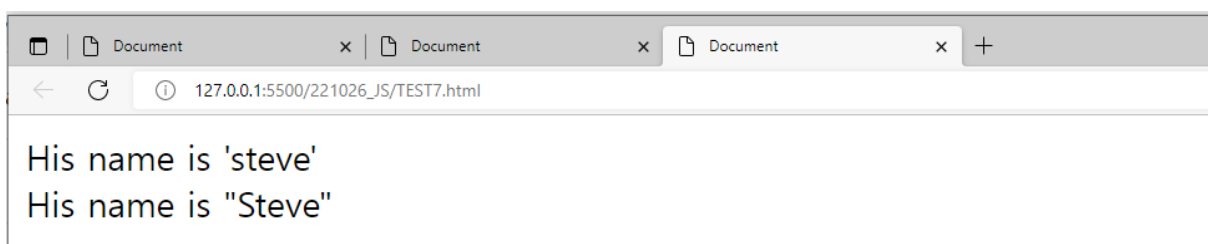
→ 'say "HI" '

TEST7.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var msg = "His name is 'steve'";
    document.write(msg + "<br>");

    var msg = 'His name is "Steve"';
    document.write(msg);

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



## 이스케이프 시퀀스 \

이스케이프 시퀀스 다음에 오는 특수문자들은 그 기능이 무시된다.

TEST7.html

```
<!DOCTYPE html>
<html>
<body>
<script>

var msg = 'His name is \'Steve\'';
document.write(msg + "<br>");

var msg = "His name is \"Steve\"";
document.write(msg);

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

\n 줄바꿈

\t 탭 띄우기

TEST8.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

  <pre>
  <script>
    var msg = 'His \t\t\t name is \'Steve\'';
    // \t\t\t가 먹히지 않는다
    document.write(msg + "\n");
  </script>
  </pre>

  <!-- pre 태그로 script전체를 감싸주면 \t가 정상적으로 작동한다. -->
</body>
</html>
```

## 산술 연산자 +, -, \*, /, ++, --, %

TEST9.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>

    var num1 = 7;
    var num2 = 3;
    var res;

    res = num1 + num2;
    document.write("num1 + num2 = ");
    document.write(res);

    res = num1 - num2;
    document.write("<br> num1 - num2 = ");
    document.write(res);

    res = num1 * num2;
    document.write("<br> num1 * num2 = ");
    document.write(res);

    res = num1 / num2;
    document.write("<br> num1 / num2 = ");
    document.write(res);

    res = num1 % num2;
    document.write("<br> num1 % num2 = ");
    document.write(res);

  </script>

</body>
</html>
```



```
num1 + num2 = 10
num1 - num2 = 4
num1 * num2 = 21
num1 / num2 = 2.3333333333333335
num1 % num2 = 1
```

TEST10.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var num1 = 7;
    var num2 = 3;

    document.write("원래의 num1 = ");
    document.write(num1);

    ++num1;
    document.write("<br> 첫번째 ++num1 = ");
    document.write(num1);

    ++num1;
    document.write("<br> 두번째 ++num1 = ");
    document.write(num1);

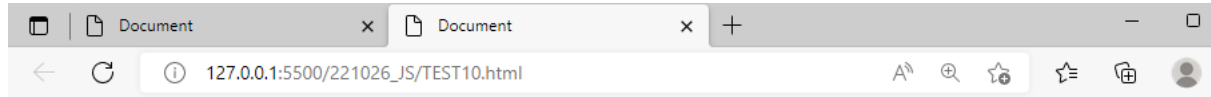
    document.write("<br><br> 원래의 num2 = ");
    document.write(num2);

    --num2;
    document.write("<br> 첫번째 num2 = ");
    document.write(num2);

    --num2;
    document.write("<br> 두번째 num2 = ");
    document.write(num2);
```

```
</script>
```

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



원래의 num1 = 7  
첫번째 ++num1 = 8  
두번째 ++num1 = 9

원래의 num2 = 3  
첫번째 num2 = 2  
두번째 num2 = 1

## 할당 연산자

연산자	사용 예	동일 식
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %/ y	x = x % y

## 비교 연산자

== 같다.

!= 아니다 (산업 표준은 <>가 맞는 표현이다.)

<, >, <=, >= 등등

JS 에서 추가되는 연산자

=== : 값 뿐만 아니라 데이터 타입, 즉 자료형까지 체크하여 비교

!== : 값 뿐만 아니라 데이터 타입, 즉 자료형까지 체크하여 비교

TEST11.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var num1 = 7;
    var num2 = 3;
    var res;

    res = (num1 == num2);
    document.write("num1 == num2 : ");
    document.write(res);

    res = (num1 != num2);
    document.write("<br> num1 != num2 : ");
    document.write(res);

    res = (num1 === "7");
    document.write('<br>num1 === "7" : ');
    document.write(res);

    res = (num1 !== "7");
    document.write('<br>num1 !== "7" : ');
    document.write(res);

    res = (num1 === 7 );
    document.write("<br> num1 === 7 : ");
    document.write(res);
  </script>
</body>
</html>
```



```
num1 == num2 : false
num1 != num2 : true
num1 === "7" : false
num1 !== "7" : true
num1 === 7 : true
```

## 논리 연산자 && || ^

연산자	기능	사용 예
&&	논리곱(and 연산. 둘 다 true이어야 결과가 true)	(x < 10 && y > 5)
	논리합(or 연산. 둘 중 하나만 true이면 결과가 true)	(x == 5    y == 10)
!	부정(not 연산. true이면 false, false이면 true)	!(x == y)

## 삼항 연산자

if 없이 if를 구현

3항 연산자는 하나의 조건을 판별하여 true와 false일 경우 각각 다른 구문을 실행한다.

기본형태

(조건식) ? true 일 경우 실행 구문 : false 일 경우 실행 구문;

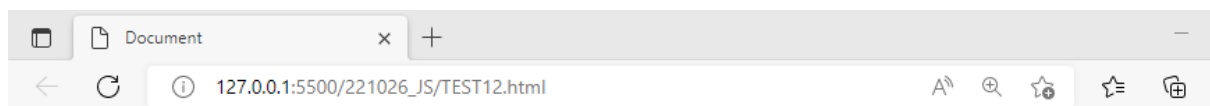
TEST12.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var num1 = 7;
    var num2 = 3;
    var op;
    var res;

    op = "+";
    res = (op == "+") ? num1+num2 : num1-num2;
    document.write("res = ");
    document.write(res);

    op = "-";
    res = (op == "+") ? num1+num2 : num1-num2;
    document.write("<br>res = ");
    document.write(res);

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



res = 10

res = 4

## 문자열 연산자

TEST13.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```

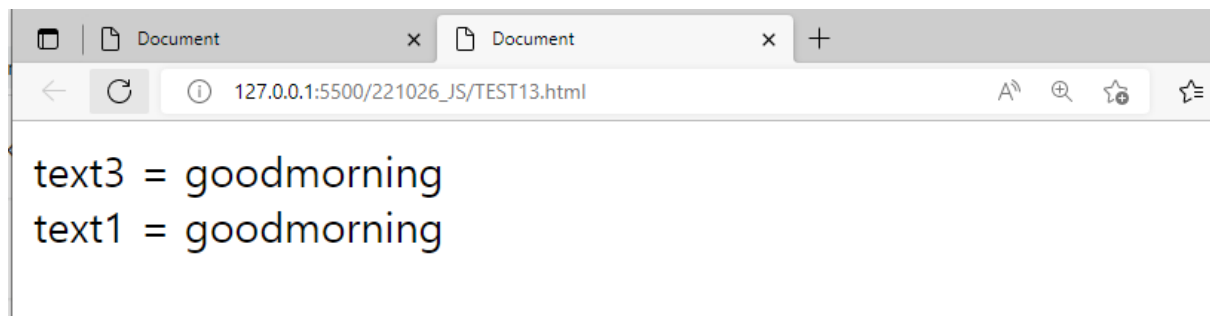
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>
  <script>
    var text1 = "good";
    var text2 = "morning";
    var text3;

    text3 = text1 + text2;
    document.write("text3 = ");
    document.write(text3);

    text1 += text2;
    document.write("<br>text1 = ");
    document.write(text1);

  </script>
</body>
</html>

```



## IF 조건문

특정 구문을 검사하여 그 결과에 따라

true 인 경우의 실행 구문 과false 인경우의 실행 구문이 달라지게 된다.

마치 컴퓨터가 사람처럼 다양한 판단을 하고 구분할 수 있도록 해주는 것이 조건문의 역할이다.

인공지능도 정교하고 복잡한 조건문 없이는 불가능하다.

따라서 조건문을 잘 이해하는 것이 프로그래밍 학습의 기본이고 시작이다.

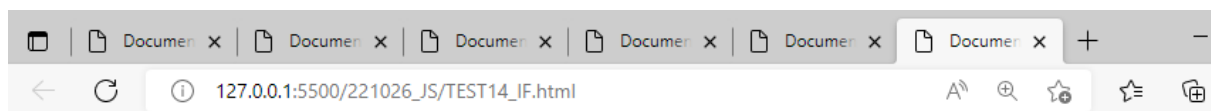
기본형태

```
if(조건식){
    조건이 참일 경우 실행 구문;
}
```

```
TEST14_IF.html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var v;
        v = prompt("점수를 입력하세요!!");

        if(v >= 90){
            document.write("A학점 입니다.<br>");
        }document.write("점수는 " + v + "점 입니다");

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



A학점 입니다.  
점수는 95점 입니다

## if else

기본형태

```
if(조건식){
```

```

    조건이 참일 경우 실행 구문;
}else{
    조건이 거짓일 경우 실행 구문;
}

```

```

TEST14_IF.html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var v;

        v = prompt("점수를 입력하세요!!");

        if(v >= 90){
            document.write("A학점 입니다.<br>");
        }else{
            document.write("B학점 입니다.");
        }

        document.write("점수는 " + v + "점 입니다");

    </script>

</body>
</html>

```

## if - else if

기본형태

```

if(조건식){
    조건이 참일 경우 실행 구문;
}else if(조건식){
    조건이 참일 경우 실행 구문;
}else if(조건식){
    조건이 참일 경우 실행 구문;
} else{

```

```
조건식이 거짓일 경우 실행 구문;  
}
```

quiz1 점수를 입력 받아서 점수에 따라 학점을 출력하세요.

90점이상 a

80점이상 b

70점이상 c

60점이상 d

나머지 f

```
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta http-equiv="X-UA-Compatible" content="IE=edge">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Document</title>  
</head>  
<body>  
  <script>  
    var n = prompt("당신의 이름을 입력하세요"); // 이름 입력  
    var v = prompt("점수를 입력하세요"); //점수 입력  
    var k; //학점  
    if(v >= 90){  
      k = "A";  
    }else if(v >= 80){  
      k = "B";  
    }else if(v >= 70){  
      k = "C";  
    }else if(v >= 60){  
      k = "D";  
    }else{  
      k = "F";  
    }    document.write(n + "님의 점수는 " + v + "점으로 " + k + "학점 입니다.");  
  
  </script>  
</body>  
</html>
```

TEST16\_SWITCH.html  
스위치 문으로 작성

```

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var name = prompt("이름을 입력하세요.");
    var x = prompt("점수를 입력하세요.");
    var z = x - (x % 10);
    var k;
    switch(z) {
      case 90:
        k = "A"
        break;
      case 80:
        k = "B"
        break;
      case 70:
        k = "C"
        break;
      case 60:
        k = "D"
        break;
      default:
        k = "F"
        break;
    }
    document.write(name + "님의 점수는 " + x + "점이므로 " + k + "학점입니다.")

  </script>
</body>
</html>

```

## 반복문 **for** while do~while break continue

기본 형태

```

for ( start : stop : step ) {

조건이 참일 동안 실행될 구문

}

```

TEST17.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    for(i=1;i<=10;i++){
      document.write(i+"<br>");
    }
    document.write("END")

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

```
1
2
3
4
5
6
7
8
9
10
END
```

TEST18.html

10까지의 합

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
```

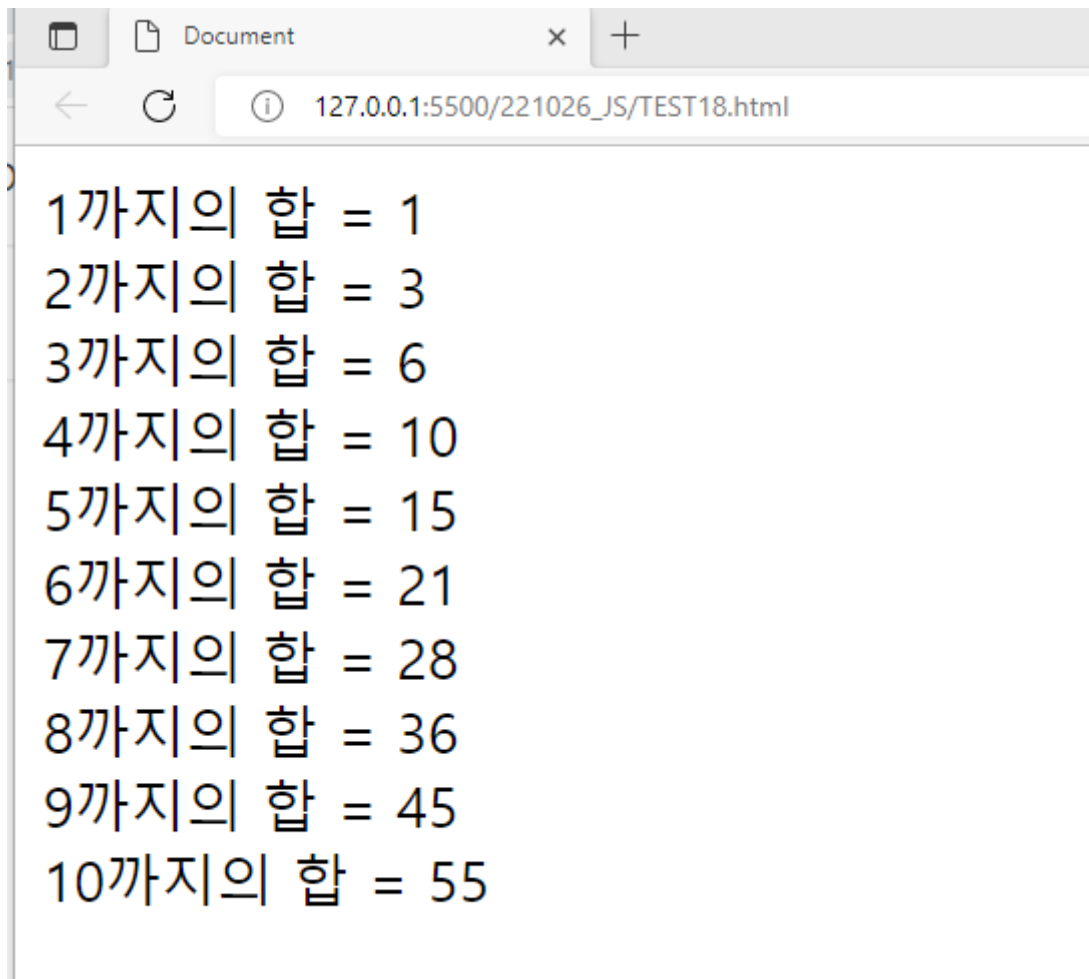


```

<body>
  <script>
    var sum = 0;
    for(i=1;i<=10;i++){
      sum += i;
      document.write(i+"까지의 합 = " + sum + "<br>");
    }

  </script>
</body>
</html>

```



```

TEST19.html
구구단

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

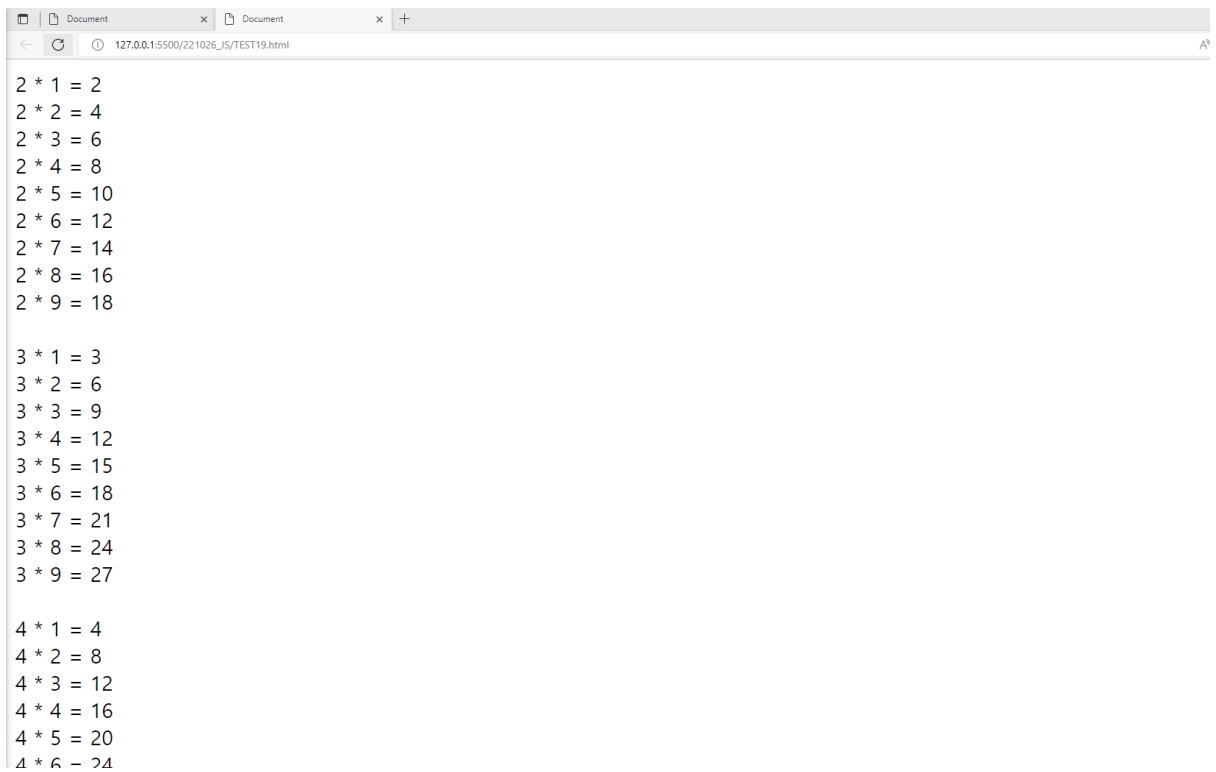
```

```

<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>
  <script>
    var sum = 0;
    for(i=2;i<=9;i++){
      for(j=1; j<=9;j++){
        document.write(i + " * " + j + " = " + (i*j) + "<br>");
      } document.write("<br>");
    }

  </script>
</body>
</html>

```



```

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27

4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24

```

구구단 가로로

```

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

```

```

<body>
  <script>
    var sum = 0;
    for(i=2;i<=9;i++){
      for(j=1; j<=9;j++){
        document.write(j + " * " + i + " = " + (i*j) + "\t" );
      } document.write("<br>");
    }

  </script>
</body>
</html>

```

## while문

for문으로 작성할 수 있는 모든것은 while문으로, while문으로 작성할 수 있는 모든것은 for문으로 작성 가능하다.

그러나, 일반적으로 반복의 횟수가 정확할 경우는 for문 , 확실치 않을 경우에는 while문을 사용한다.

```

기본형태
while(조건문){
  조건을 만족하는 경우 실행되는 구문;
}

```

TEST20.html - while문으로 구구단 만들기

```

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var dan = 2;

    while (dan <= 9){

      var num = 1;
      while(num <= 9){
        document.write(dan + " * " + num + " = " + (dan*num) + " ");
      }
    }
  </script>

```

```

        num++;
    }
    dan++;
    document.write("<br>");
}
</script>
</body>
</html>

```

## JavaScript 난수 발생

Math.random(); : 난수를 발생 시키는데 0에서 1미만 구간에서 난수를 발생시킨다.

```

TEST21.html

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    // 1단계 기본 0 ~ 1 미만의 난수를 발생 시킨다.
    ran1 = Math.random();

    document.write(ran1 + "<br>");

    //2단계 Math.floor 매서드를 적용
    // Math.floor 메서드는 소수점 1번째 자리를 버림하여 정수를 리턴한다.
    ran2 = Math.floor(Math.random() * 10);

    document.write(ran2 + "<br>");

    //3단계 소수점 2번째 자리를 버림하여 정수 리턴
    ran3 = Math.floor(Math.random() * 100);

    document.write(ran3 + "<br>");

  </script>
</body>
</html>

```

## 스무고개 게임 만들기

TEST22.html - 내가 만든것

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

  <h1>20고개 게임을 시작합니다.</h1>
  <button onclick="game()">게임 시작</button>
  <script>
    function game(){

      var x = Math.floor(Math.random() * 100);
      var i = prompt("1 ~ 100중 한개의 숫자를 입력하세요.");
      var z = 1;

      while ( i < 100){
        if(i<x){
          i = prompt("더 큰수를 입력하세요");
          z++;
        }else if(i > x){
          i = prompt("더 작은수를 입력하세요.");
          z++;
        }else{
          document.write("정답입니다.");
          document.write(z + "번째에 맞추셨습니다.")
          break;
        }
      }

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

선생님이 만든 것

```
<h2>20고개 게임을 시작합니다. </h2>

  <button onclick="game()">게임 시작</button>

  <script>
    function game() {
```

```

var answer = Math.floor(Math.random() * 100);
var no = 0;
var count = 0;
while(answer != no) {
    var no = prompt("숫자를 입력하세요");

    if(no > answer) {
        alert("더 작은 수를 입력하세요 !!!")
    }else {
        alert("더 큰수 수를 입력하세요 !!!")
    }
    count++;
}

document.write(count + "번 만에 성공하였습니다. ");
}
</script>

```

## alert

```

alert(" 알리고 싶은 말");

```

## do ~while

while과의 차이점은 반드시 한번은 꼭 실행한다는 것이다.

```

do {
    실행문;
}while(조건식);

```

※ 쉽게 생각하면 while의 조건식이 true 이면 do의 실행문이 반복되고, while의 조건식이 false이면 탈출!

TEST23.html - 숫자를 하나 입력 받아서 그 숫자의 구구단을 출력하세요.

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>구구단 출력기</h1>
  <button onclick="game()">시작</button>
  <script>
    function game()
    {
      var i = prompt("숫자를 입력하세요.");
      var j = 1;
      if(i>9){
        do {
          i = prompt("숫자가 큼니다. 1-9 사이의 숫자를 입력하세요");
        }while(i>9)
      }

      do{
        document.write(i + " * " + j + " = " + (i*j) + "<br>");
        j++;
      }while(j <10)

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

※do ~while 더 공부할 것