7_11_2022_js_-loop, if, createElement, innerHTml, operator

```
Javascript Step02.1_loop01.html
            ~ stylesheet  Step17 css.html


                                    {("string type :event name", call back function)}
    document.querySelector("#myBtn").addEventListener("click", function(){
        console.log("clicked2()호출됨")

For
     for(let i=0; i<5; i++){

        1.let i=0 는 for 문 안에서만 사용할 수 있는 local variable 를 만들고 초기값 0 대입
          반복문이 수행되는 동안 유지된다.
          for 문이 종료되는 i 는 사라진다.
          for 문에 실행순서가 왔을 때 최초 한번만 실행된다.

        2. i<5 이 조건이 false 가 될 때 까지 for 문의{} 블럭이 반복 수행된다.
        3. i++ 에서 ++는 i 안에 있는 숫자를 1 증가 시키는 증감연산자 이다.

    for문은 배열에 많이 사용한다.
        let nums=[10,20,30,40,50];
                i 는 제어변수
    for(let i=0; i<nums.length; i++){
        i번째 방에 들어있는 item 참조하기
        let tmp=nums[i];
            참조된 값 사용하기
        console.log(tmp);
        }

Thread
    순서대로 실행되는 작업의 흐름
    함수를 만들어서 전달만 해 놓으면 적정한 시점에 알아서 호출 됨.]

if
    제어문, 조건부 수행
        if(){}
            true가 참조되면 실행됨 false면 빠져나감
            코드블럭을 조건부로 수행하기

        if (){
        }else{}
```

```
            두개의 블럭을 가져다 놓고 양자택일을 할 수 있다.
            ()에 true가 참조되면 if에 작업을 수 행하고 빠져나간다.
            false가 참조되면 else뒤의 코드블럭이 실행되고 빠져나간다.
            양자택일
            ifelse문을 겹쳐서 여러가지 블럭의 true false 참조한다.

반복문 for 조건문 if 많이 사용됨.


연산자
    1. 산술연산자
        number type 데이터를 연산할때 사용한다.
        +, -, *, /, % (나머지 연산자)
            let result1=10+1; //11
            let result2=10-1; //9
            let result3=10*10; //100
            let result4=10/2; //5
            let result5=10%3; //1

            let num1=10;
            let num2=1;
        산술연산을 할때 숫자가 들어있는 변수명으로 할수도 있다.
            let result6=num1+1;
            let result7=num1%3;
            let result8=num1-num2;

        object 혹은 array 에 들어 있는 데이터를 이용해서 연산을 할수도 있다.
            let obj={first:10, second:20};
            let arr=[10, 20, 30];

            let result9=1+obj.first;
            let result10=10*arr[1];

    2. 논리(boolean) 연산자
        - 논리값을 연산해주는 연산자

        -&& 는 and 연산자이고 읽을때는 "그리고" 라고 읽으면 된다.
            연산의 법칙은 연산에 참여하는 모든 boolean type 데이터가
            true 일때만 결과값이 true 가 나온다.
                만일 inputNum 이 0 보다 크고(그리고) inputNum 을 2 로 나눈 나머지가 0 과 같다면
                    if(inputNum>0 && inputNum%2==0){
                 console.log("입력한 수는 0보다 크고 동시에 짝수 입니다.");}\

        - || 는 or 연산자이고 읽을때는 "또는" 이라고 읽으면 된다.
            연산의 법칙은 연산에 참여하는 모든 boolean type 데이터가
            어느 하나만 true 이면 결과는 true 가 나온다.
                만일 inputNum 이 0 보다 크거나 inputNum 을 2로 나눈 나머지가 0과 같다면
                    if(inputNum>0 || inputNum%2==0){
                    console.log("입력한 수는 0보다 크거나 또는 짝수 입니다.");}


                        let result1=false&&false; //false
                        let result2=false&&true; //false
                        let result3=true&&false; //false
                         let result4=true&&true; //true

                        연산에 참여하는 boolean 값이 어느 하나만 true  면 결과는 true 이다.
                        let result5=false||false; //false
```

```
                        let result6=false||true; //true
                        let result7=true||false; //true
                        let result8=true||true; //true


        -! 는 논리값을 반전시키는 not 연산자이다.
            읽을때는 "~가 아니면" 이라고 읽으면 된다.
                1. 만일 isRun 이 false 라면
                2. 만일 isRun 이 true 가 아니라면
                3. 만일 달리지 않을꺼면
                     if( !isRun ){
                    }
                    if( isRun == false){
                만일 isRun 이 true 가 아니거나 isWait 이 true 가 아니라면
                만일 달리지 않거나 기다리지 않을거면
                    if(!isRun || !isWait){

        -대입 연산자.
            =   => 우측에 있는 값을 좌측에 대입
            +=  => 우측에 있는 값을 좌측에 있는 원래 값에 더해서 대입
            -=  => 우측에 있는 값을 좌측에 있는 원래 값에 빼서 대입
            *=  => 우측에 있는 값을 좌측에 있는 원래 값에 곱해서 대입
            /=  => 우측에 있는 값을 좌측에 있는 원래 값에 나누어서 대입
            %=  => 우측에 있는 값을 좌측에 있는 원래 값을 나눈 나머지 값을 대입
        -비교 연산자
            == => 같은지 비교
            != => 다른지 비교
            >  => 큰지 비교
            >= => 크거나 같은지 비교
            <  => 작은지 비교
            <= => 작거나 같은지 비교

            - 비교 연산자로 주로 하는 작업
            1. 문자열의 내용이 같은지 다른지 비교
            2. 숫자의 크기 비교
            3. 숫자가 같은지 다른지 비교
            4. 논리값이 같은지 다른지 비교
            5. 문자열의 알파벳 순서 혹은 가나다 순서 비교

        -증감 연산자.
            ++ => 1 증가 시키기
            -- => 1 감소 시키기
                증감연산자를 변수 뒤에 붙이면 연산의 우선 순위가 가장 뒤쳐진다(나중에 연산된다.)

create element
    createElement함수 element 만듦
    append 만들어진 element를 추가함
    예약어 this
        xxx.addEventListener("event name", function(){
         this <== event가 일어난 그 요소
        });


insertAdjacentHTML
    어느 구간에 입력할 것 인지 선택
        deforestart
    <div>
        afterstart
```

```
    <p>XXX</p>
        deforeend
    </div>
        afterend

    innerText와 innerHTML
        innerText는 string으로 출력
        innerHTML은 HTML로 해석해서 출력(기능을 사용 가능)

css
    외부 내부 인라인 css들이 있고 따로따로 입력해도 같이 나올 수 있다.
    색상 16진수 10진수 표현이 가능함
        16진수 #로 시작해 00부터 까지 rgb 세 값이 함께 나옴 #ffffff
        10진수 rbb(255,255,255)
    한 줄로 정리할 수 있다.
    동일한 element에 css를 주면 더 구체적인 속성이 적용된다.
    더 구체적으로 만드는 방법이 있는데 속성들을 연결해서 사용하면 된다.
        div.animals   등

    띄어쓰기로는 자손 요소를 선택 할 수 있다.
        #one li                  .animal p 등

    꺾쇠>로는 자식 요소를 선택할 수 있다.
        .congainer> p               .sub>p

    이 모든 선택자를 document.querySelector("xxxx")   xxxx에 쓸 쑤 있다.

    * 모두 선택
    , 다중 선택

    하나의 요소가 여러개의 class를 가지고 있을 수 있다. 미리 가져와서 사용.

    margin
        경계선과 바깥쪽 여백                    top
    padding                                 left      right
        경계선과 컨텐츠 안쪽 여백              bottom
    border                                              top 부터 시계방향으로 기억하기
        경계선

    사용법
        margin{10px 10px 10px 10px 한번에 사용할 수 있다.} top부터 시계방향으로
        margin{10px 20px} top,bottom 10px   right,left 20px
        margin{10px 20px 30px} top 10px right 20px bottom 30px
        padding도 대동소이

        border는 세가지 속성이 있다. width, style, color
            각각 방향과 속성을 다르게 적용할 수 있다.'

position
    static  문서기준으로 정적인 - default값
    relative 상대적인 원 위치를 기준으로 왼쪽으로 부터 얼마 위에서 부터 얼마 등
    absolute 절대적인  특정 요소를 기준으로 절대 좌표에 배치
    fixed   윈도우 기준으로 절대 좌표에 배치할 때 사용 스크롤 해도 그 자리에 있다


Pseudo class (가상 클래스)
    클래스로 지정해주지 않아도 가상으로 추가되는 클래스로
        .box:hover{}     hover 라는 가상 클래스
```

```
        .input:focus{}   focus 라는 가상 클래스
        button:active{}  active라는 가상 클래스로
            link, visited 등등..


float 띄우기 속성
    block element도 필요한 만큼만 차지하게 만들 수 있다.
    p요소 위에 떠있는 것 처럼 자리를 차지하고 있다.
    안의 내용은 따로 어우러지도록 배치가 된다.
    block element를 inline element 처럼 만들 수 있다.
    크기를 정하고 float속성을 주면 됨.
    크기를 지정하지 않으면 필요한 만큼만 크기가 정해진다.
    clear속성을 추가하면 깔끔하게 정리할 수 있다. left|right|both
    overflow:auto 를 사용하면 감싸진다.
    top, right, bottom, left를 0으로 만드면 전체를 채운다.

display
    none 안보이게 하기
    inline, block 보이게 하기
    화면 전체를 덮는 div를 backdrop이라고 부른다.
    inline 요소가
        width와 heigh를 지정할 수 없다.
        left, right margin, padding 을 가질  수 없다.
        top, bottom margin, padding을 가질 수 없다.
        왼쪽과 오른쪽에 다른 요소가 올 수 있다.
    inline block 요소
        width, height, margin, padding 모두 가능
        왼쪽과 오른쪽에 다른 요소가 올 수 있다.

forEach
    변형된 반복문

z index
    z index가 크면 클 수록 가장 위에 올라온다.
    순서를 조절할 수 있다.
    x축 y축 그리고 나를 향해 오는 값이 z축

text
    글자에 관련된 요소는 상속된다.
        부모요소에 적용해도 일괄적으로 적용된다.
    align 정렬 left center right
        단어 하나하나는 inline요소 처럼 취급됨.
    justify 공간 늘려 창에 꽉 채우기.

가중치
    가중치가 같다면 나중에 정의한 것이 적용된다.
        !important          *****
        inline css          ****
        id                  ***
        class               **
        css                 *
    구체적으로 정의되면 같은 속성보다 가중치가 크다
        li.id               ***+
        id                  ***
        li.class            **+
        class               **

visability
```

```
    투명도
overflow
    부모컨텐츠보다 자식 컨텐츠가 클 수도 있다.
    default는 그냥 보이게 하는 것 삐져나옴
    hidden 숨긴다.
    scroll 스크롤해서 보이게 만듦
    auto 넘어가지 않으면 스크롤바가 없고 넘어가면 생김.

bixsizing
    padding과  border를 width height에 포함시키면어떨까?
    계산이 편리해짐
        content-box (default) | border-box
        content-box : width 와 height 에 아무것도 포함하지 않음
        border-box : width 와 height 에 padding 과 border 포함

크기
   [ 크기나 거리를 나타내는 단위 ]

    px, %, em, rem

    px => 점(화소)의 갯수
    % => 부모의 크기에 대한 상대적 비율
    em => 상속받은 글자의 크기의 배수    em은 그때그때 기준이 바뀜
    ex) 상속받은 글자의 크기가 16px 이라면

    1em 은 16px
    1.5em 은 24px
    2em 은 32px
    3em 은 48px

    rem => 최상위요소(html) 의 글자의 크기의 배수
    html 의 font-size: 14px 이라면
    1rem => 상속받은 글자의 크기와 상관없이 14px
    2rem => 상속받은 글자의 크기와 상관없이 28px
    최근에는 em을 많이 사용하지 않고 rem을 사용함 root 글자크기 바꾸면 다 바뀜
        1 rem 의 기준 크기 (default 값은 16px)
            최상위(root) 글자 크기의 배수
            html 요소의 글자 크기의 배수 이다.
```