인포그래픽 제작을 위한
JS + Chart.js + CSS 작성법

CONTENTS

1. HTML 네이밍 룰

Pure JS 함수를 통해 애니메이션 차트를 구현하기 위해서는 선택 아닌 필수

차트의 값에 해당하는 요소 – Value(abbr. Val),
차트 구성에 따라 e.g. Head/Body/Foot 등 하나의 명칭을 정한 후
{영역명}{명칭}{넘버링}의 형식으로 id를 선언하는 것을 권장

```
<c:forEach var="section7" items="${ rnRank10SumAnually }" begin="0" end="11" step="1" varStatus="status">
    <div class="rn-chart-divider${ status.count }">
        <p class="rn-datalabel" id="section7Val${ status.count }" name="${ section7.val }"><fmt:formatNumber value="${ section7.val }" pattern="0.0"/></p>
        <img src="/resources/images/visualization/infographics/wave2_01.png" class="anl-rain-sum-head" id="section7Head${ status.count }"
             alt="이미지를 불러올 수 없습니다."/>
        <div class="rn-body" id="section7Body${ status.count }">
            <div class="rn-rank-divider" id="rnRankBody${ status.count }">
                <span class="rn-rank">${ section7.ranking }</span><span class="rn-rank-unit">위</span>
            </div>
        </div>
        <p class="ta-rn-stn">${ section7.stnNm }</p>
        <p class="ta-rn-value">${ section7.stnId }</p>
    </div>
</c:forEach>
```

1. HTML 네이밍 룰

명칭이 원칙에 따라 통일되어 있어야 차트 애니메이션에 파라미터를 대입할 때 보다 가독성 좋은, 통일된 양식의 함수를 작성할 수 있음

```
function tooltipsHandler(targetSuffix, objectSuffix, i){
    var targetHead = "#section" + targetSuffix + "Head" + i;
    var targetBody = "#section" + targetSuffix + "Body" + i;

    var objectHead = "#section" + objectSuffix + "Head" + i;
    var objectBody = "#section" + objectSuffix + "Body" + i;

    $(objectHead).css("display", "inline-block").css("display", "none");
    $(objectBody).css("display", "inline-block").css("display", "none");

    $(targetBody).mouseenter(function(){
        $(objectHead).css("display", "inline-block").css("position", "absolute");
        $(objectBody).css("display", "inline-block").css("position", "absolute");
    });

    $(targetBody).mouseleave(function(){
        $(objectHead).css("display", "inline-block").css("display", "none");
        $(objectBody).css("display", "inline-block").css("display", "none");
    });
}
```

2. calc

반응형/동적 애니메이션을 작성하기 위한 필수이자 핵심 CSS 문법

앞으로 설명할 애니메이션 함수는

공식에 따라 계산된 값을 calc 문법에 대입하여 처리하는 것이 핵심 원리

```
if(bodyPositionPoint < bodyReachPoint){
    if(headPositionPoint < headReachPoint && head != null){ head.style.bottom = headPositionPoint + "%"; }
    body.style.bottom = "calc(0 + (" + headHeight + "px / " + positionDivider + ")";
    body.style.height = "calc("+ bodyPositionPoint + "% - (" + headHeight + "px / " + positionDivider + "))";
```

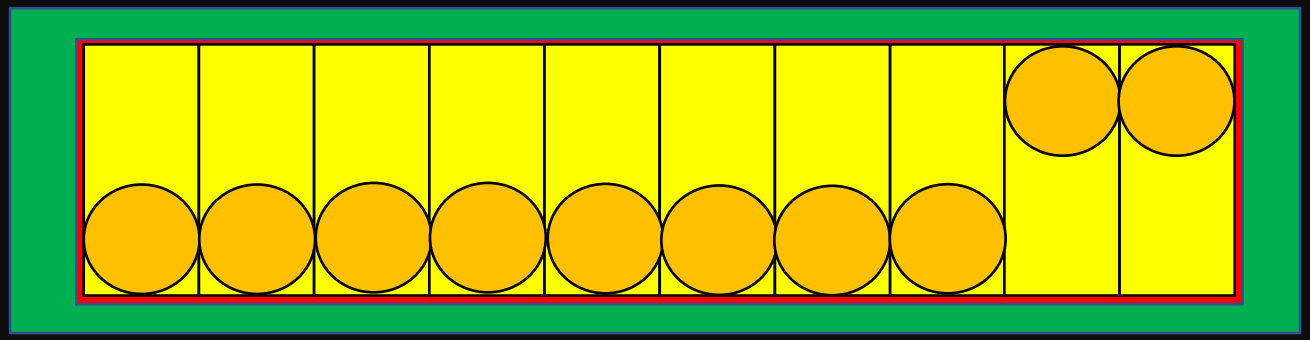
```
#section2Head1{ margin-left: -7.4%; }
#section2Head2 { margin-left: calc(-7.4% + (8.48% * 1)); }
#section2Head3 { margin-left: calc(-7.4% + (8.48% * 2)); }
#section2Head4 { margin-left: calc(-7.4% + (8.48% * 3)); }
#section2Head5 { margin-left: calc(-7.4% + (8.48% * 4)); }
#section2Head6 { margin-left: calc(-7.4% + (8.48% * 5)); }
#section2Head7 { margin-left: calc(-7.4% + (8.48% * 6)); }
#section2Head8 { margin-left: calc(-7.4% + (8.48% * 7)); }
#section2Head9 { margin-left: calc(-7.4% + (8.48% * 8)); }
#section2Head10 { margin-left: calc(-7.4% + (8.48% * 9)); }
#section2Head11 { margin-left: calc(-7.4% + (8.48% * 10)); }
#section2Head12 { margin-left: calc(-7.4% + (8.48% * 11)); }
```

```
.mnh-rain-wrapper1, .rn-10y-labels1 { margin-left: calc((58px * 0) + 13px ); }
.mnh-rain-wrapper2, .rn-10y-labels2 { margin-left: calc((58px * 1) + 13px ); }
.mnh-rain-wrapper3, .rn-10y-labels3 { margin-left: calc((58px * 2) + 13px ); }
.mnh-rain-wrapper4, .rn-10y-labels4 { margin-left: calc((58px * 3) + 13px ); }
.mnh-rain-wrapper5, .rn-10y-labels5 { margin-left: calc((58px * 4) + 13px ); }
.mnh-rain-wrapper6, .rn-10y-labels6 { margin-left: calc((58px * 5) + 13px ); }
.mnh-rain-wrapper7, .rn-10y-labels7 { margin-left: calc((58px * 6) + 13px ); }
.mnh-rain-wrapper8, .rn-10y-labels8 { margin-left: calc((58px * 7) + 13px ); }
.mnh-rain-wrapper9, .rn-10y-labels9 { margin-left: calc((58px * 8) + 13px ); }
.mnh-rain-wrapper10, .rn-10y-labels10 { margin-left: calc((58px * 9) + 13px ); }
.mnh-rain-wrapper11, .rn-10y-labels11 { margin-left: calc((58px * 10) + 13px ); }
.mnh-rain-wrapper12, .rn-10y-labels12 { margin-left: calc((58px * 11) + 13px ); }
```

3. HTML formatting

효과적인 함수 대입 및 균일한 영역 처리를 위해 divider 구조로 작성을 권장

Top: 0
/ position:
absolute /
Bottom: 0



section
divider
aligner
element

```
<section class="ta-rn-section3">
    <p class="theYear" id="ta-section3-year">2020</p>
    <div class="ta-section3-chart" id="taSecThree">
        <c:forEach var="section3" items="${ taRank5MaxAnually }" begin="0" end="4" step="1" varStatus="status">
            <div class="ta-chart-divider${ status.count }">
                <div class="max-bar-aligner">
                    <p class="ta-max-datalabel" id="section3Val${ status.count }" name="${ section3.val }"><fmt:formatNumber value="${ section3.val }" pattern="0.0"/></p>
                    <div class="max-bar" id="section3Elem${ status.count }"></div>
                    <div class="max-barFoot"></div>
                </div>
                <div class="max-circle"><span class="ta-rank">${ section3.ranking }<span class="ta-rank-unit">위</span></span></div>
                <p class="ta-rn-stn">${ section3.stnNm }</p>
                <p class="ta-rn-value">${ section3.stnId }</p></p>
            </div>
        </c:forEach>
    </div>
</section>
```

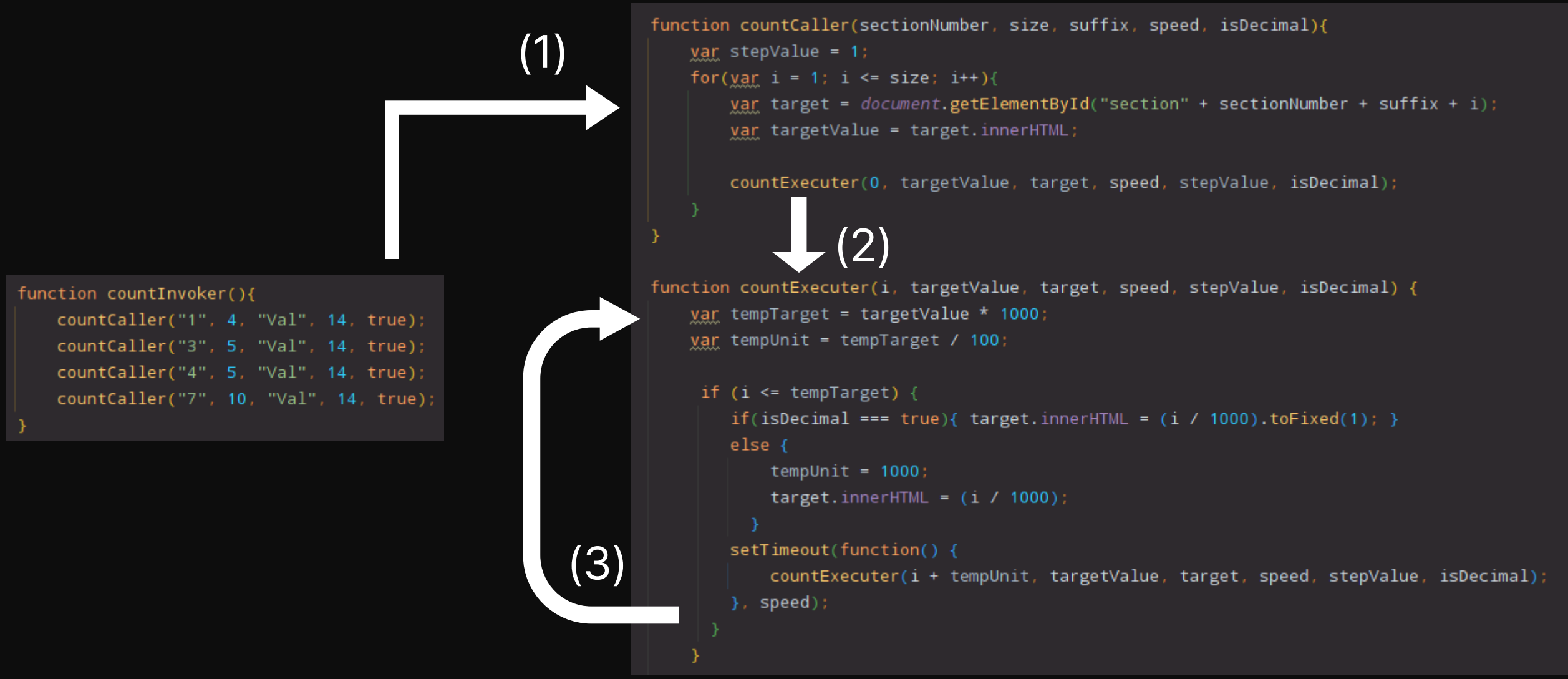
4. 함수 분리

Java라고 생각하자. 원리는 같음.

단, 복수의 요소를 다루는 경우 아래와 같은 형식 등으로 구조화 하는 것을 권장
e.g. Invoker (파라미터 대입) / caller (로직 호출) / executor (비즈니스 로직)

Chart.js 설정/생성 구문도, JS 함수도, CSS 대입문도 모두 함수로 분리 가능
요소 수가 많으면 많을수록 함수 분리는 선택이 아닌 필수

4. 함수 분리




```
function countInvoker(){
    countCaller("1", 4, "Val", 14, true);
    countCaller("3", 5, "Val", 14, true);
    countCaller("4", 5, "Val", 14, true);
    countCaller("7", 10, "Val", 14, true);
}
```

```
function countCaller(sectionNumber, size, suffix, speed, isDecimal){
    var stepValue = 1;
    for(var i = 1; i <= size; i++){
        var target = document.getElementById("section" + sectionNumber + suffix + i);
        var targetValue = target.innerHTML;

        countExecuter(0, targetValue, target, speed, stepValue, isDecimal);
    }
}

function countExecuter(i, targetValue, target, speed, stepValue, isDecimal) {
    var tempTarget = targetValue * 1000;
    var tempUnit = tempTarget / 100;

     if (i <= tempTarget) {
        if(isDecimal === true){ target.innerHTML = (i / 1000).toFixed(1); }
        else {
            tempUnit = 1000;
            target.innerHTML = (i / 1000);
          }
        setTimeout(function() {
            countExecuter(i + tempUnit, targetValue, target, speed, stepValue, isDecimal);
        }, speed);
     }
    }
```

5. count

[개요]

0부터 설정된 목적 값까지 순차적으로 값을 증가시키는 JS 애니메이션 함수

[구성]

countInvoker / countCaller / countExecutor

[대입 파라미터]

{ 섹션 넘버 } { 요소 수 } { id후위명칭 } { 속도 } { 소수여부 }

5. count

[FLOW]
/ invoker
0. 파라미터 전달

/ caller
1. 네이밍 룰에 따라 작성된 섹션 넘버링을 통해 대상 요소를 특정
2. 특정된 Value 요소의 innerHTML을 목표 값으로 설정

/ executor
3. 소수연산 issue를 피하기 위해 연산 단위를 조정
4. 0부터 시작하여 목표 값에 도달할 때까지 값을 더하는 재귀함수 실행

6. raise

[개요]

Head/Body로 구성된 동적 2중 차트 구현을 위한 순수 JS 애니메이션 함수

[구성]

raiseInvoker / raiseCaller / raiseExecutor

[대입 파라미터]

{ 섹션 넘버 } { Head-Id 후위명칭 } { Body-id 후위명칭 } { 시작 index } { 끝 index }

{ Head 위치 } { 최소값 } { 최대범위값 } { Head 높이 } { Body 기본값 } { positionDivider } { 속도 }

6. raise

[FLOW]
/ invoker
0. 파라미터 전달

/ caller
1. 네이밍 룰에 따라 작성된 섹션 넘버/후위 명칭을 통해 대상 요소를 특정
2. 특정된 Value/Elem 요소의 innerHTML을 목표 값으로 설정
3. 목표 값의 단위 절삭, 최소값과 최대값 연산을 통해 순수 목표 값을 설정
4. 목표 값의 조건부 후처리, 최대 한계 값을 설정

/ executor
3. Head 목표 값을 설정 = (목표값/최대값) * 최대범위값;
4. Body 목표 값을 설정 = (목표값/최대값) * 최대범위값 + Body기본값;
5. Head가 null인 경우 Head 애니메이션을 생략, null이 아닌 경우 bottom기준 %단위로 높이 설정 bottom = headPositionPercent%
6. Body를 bottom기준 %단위로 높이 설정 calc(0 + (Head높이px / positionDivider));
7. Body를 bottom기준 %단위로 길이 설정 calc(bodyPositionPoint% - (headHeightpx / positionDivier));
8. 최종 목표 값에 도달할 때까지 5~ 7과정을 반복하는 재귀함수 실행

7. Chart.js

```
function chartsAssigner(){
//  canvas, label, data, pointRadius, borderColor, borderWidth, pointType, minValue, maxValue
    chartGenerator(section2Max, noLabel12, dataSpreader()[0][0], 3, maxTempColor, 2, thermometerMax, -10, 35.0);
    chartGenerator(section2Min, noLabel12, dataSpreader()[0][1], 3, minTempColor, 2, thermometerMin, -10, 35.0);
    chartGenerator(section2Avg, noLabel12, dataSpreader()[0][2], 3, avgTempColor, 2, thermometerAvg, -10, 35.0);

    chartGenerator(section2MaxNmyr, noLabel12, dataSpreader()[1][0], 5, maxTempNmyrColor, 2, "rect", -10, 35.0);
    chartGenerator(section2MinNmyr, noLabel12, dataSpreader()[1][1], 5, minTempNmyrColor, 2, "rect", -10, 35.0);
    chartGenerator(section2AvgNmyr, noLabel12, dataSpreader()[1][2], 5, avgTempNmyrColor, 2, "rect", -10, 35.0);

    chartGenerator(section5Max, dataSpreader()[2][0], dataSpreader()[2][1], 3, maxTempColor, 2, thermometerMax, 0.0, 25.0);
    chartGenerator(section5Min, dataSpreader()[2][0], dataSpreader()[2][2], 3, minTempColor, 2, thermometerMin, 0.0, 25.0);
    chartGenerator(section5Avg, dataSpreader()[2][0], dataSpreader()[2][3], 3, avgTempColor, 2, thermometerAvg, 0.0, 25.0);

    chartGenerator(section5MaxNmyr, noLabel10, dataSpreader()[3][0], 0, maxTempNmyrColor, 2, null, 0.0, 25.0);
    chartGenerator(section5MinNmyr, noLabel10, dataSpreader()[3][1], 0, minTempNmyrColor, 2, null, 0.0, 25.0);
    chartGenerator(section5AvgNmyr, noLabel10, dataSpreader()[3][2], 0, avgTempNmyrColor, 2, null, 0.0, 25.0);
}
</script>
```