7. 레이아웃

1) 속성 - display

- display 속성 : 요소의 렌더링 박스 유형을 결정하는 속성입니다.

기본값 : -(요소마다 다름)

display: value;

<속성값>

none : 요소가 렌더링 되지 않음

inline : inline level 요소처럼 렌더링

block : block level 요소처럼 렌더링

inline-block : inline level 요소처럼 렌더링(배치)되지만 block level의 성질을 가짐

* height 나 width 등과 같은 박스모델 속성을 적용할 수 있다

그외에 list-item, flex, inline-flex, table, table-cell 등 다양한 속성값 존재

inline level 요소 사이의 공백과 개행 처리 inline 요소의 경우 공백과 개행에 대해서 하나의 여백으로 받아들입니다. 따라서 inline와 inline-block의 경우 태그 사이의 공백이나 개행이 있을 경우 약 4px의 여백을 가지게 됩니다.

- display와 box model의 관계

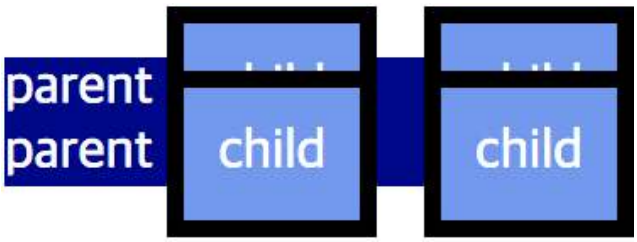
display	width	height	margin	padding	border
block	o	o	o	o	o
inline	x	x	좌/우	o(설명)	o(설명)
inline-block	o	o	o	o	o

- inline 요소의 padding/border 속성이 좌/우 만 적용 된다고 표시한 이유 추가 설명

실제로 inline 요소의 padding/border는 좌/우뿐만 아니라 상/하에도 적용이 됩니다.

parent는 , child는

하지만, 상/하 padding/border는 line-box에는 영향을 주지 못하기 때문에 위와 같이 부모 요소의 박스에 반영되지 않습니다.



parent는 , child는

또한 인접한 다른 line-box 에도 반영되지않습니다. 즉 콘텐츠가 겹칠 수 있기 때문에 실무에서는 잘 사용하지 않습니다.

<코드실습>

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>display</title>
  <style>
```

```
    body>div {
      padding: 5px;
      border: 1px dashed #aaa;
    }

    .box {
      padding: 15px;
      background-color: #eee;
      border: 1px solid #aaa;
    }

    .none .box {
      display: none;
    }

    .inline .box {
      display: inline;
    }

    .block .box {
      display: block;
    }

    .inline-block .box {
      display: inline-block;
    }

    .list-item .box {
      display: list-item;
    }
  </style>
</head>

<body>
  <h1>display</h1>
  <h2>none</h2>
  <div class="none">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
  </div>
  <h2>inline</h2>
  <div class="inline">
    <div class="box">box1</div>
```

```
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box4</div>
    <div class="box">box5</div>
    <div class="box">box6</div>
    <div class="box">box7</div>
    <div class="box">box8</div>
    <div class="box">box9</div>
    <div class="box">box10</div>
  </div>
  <h2>block</h2>
  <div class="block">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
  </div>
  <h2>inline-block</h2>
  <div class="inline-block">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
  </div>
  <h2>list-item</h2>
  <div class="list-item">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
  </div>
</body>
</html>
```

https://www.w3schools.com/cssref/pr_class_display.asp
https://developer.mozilla.org/en-US/docs/Web/CSS/display

2) 속성 - visibility

- visibility 속성 : 요소의 화면 표시 여부를 지정하는 속성입니다.

기본값 : visible

```
visibility: visible | hidden | collapse | initial | inherit;
```

<속성값>

visible : 화면에 표시

hidden : 화면에 표시되지 않음(공간은 차지함)

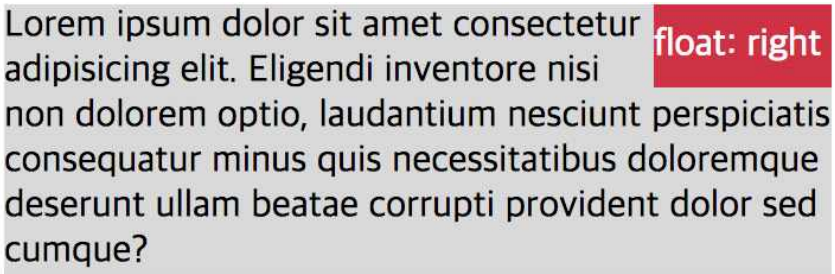
collapse : 셀 간의 경계를 무시하고 숨김(테이블 관련 요소에만 적용 가능)

```
visibility: visible;  /* 보임 기본값 */
visibility: hidden;  /* 숨김, 자신의 박스 영역은 유지(margin까지 모두 포함) */
visibility: collapse; /* 셀간의 경계를 무시하고 숨김(박스영역 없음, 테이블의 행과 열 요소에만 지정 가
```

능, 그 외 요소 지정은 hidden과 같음) */
 - display: none과 차이점
 display: none: 요소가 렌더링 되지 않음(DOM에 존재하지 않음)
 visibility: hidden: 요소가 보이지는 않지만 렌더링 되며 화면에 공간을 가지고 있음(DOM에 존재함)
<코드실습> https://www.w3schools.com/cssref/playit.asp?filename=playcss_visibility_collapse
 https://www.w3schools.com/cssref/pr_class_visibility.asp
 https://developer.mozilla.org/en-US/docs/Web/CSS/visibility

3) 속성 - float

 - float 속성 : 요소를 float(요소를 보통의 흐름에서 벗어나게 함) 시킬지 지정하는 속성입니다.
기본값 : none
float: none | left | right | initial | inherit;
 <속성값>
none : float 시키지 않음(기본값)
left : 좌측으로 float 시킴
right : 우측으로 float 시킴
 요소를 보통의 흐름에서 벗어나 띄어지게 함
 주변 텍스트나 인라인 요소가 주위를 감싸는 특징이 있음



 대부분 요소의 display 값을 block으로 변경함 (display 값 변경 예외: inline-table, flex 등)

<코드실습>
```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>float</title>
  <style>
.container {
  border: 1px dashed #aaa;
  padding: 15px;
  clear: both;
}
.container div, .container span {
  width: 100px;
```

```
  height: 100px;
  border: 1px solid #aaa;
  color: #fff;
}
.container :nth-child(1) {
  background-color: green;
}
.container :nth-child(2) {
  background-color: skyblue;
}


</style>
</head>
<body>
  <h2>요소를 보통의 흐름에서 벗어나 띄워지게 함</h2>

  <div class="container" style="width:400px;">
    <div style="float:left;">Box1</div>
    <div style="float:left;">Box2</div>
  </div>

  <h2 style="margin-top:100px;">주변 텍스트나 인라인요소가 주위를 감싸는 특징이 있음.</h2>
  <div class="container" style="width:400px;">
    <div style="float:left;">Box1</div>
    <div style="float:right;">Box2</div>
    <p>CSS 속성(property) float 은 한 요소(element)가 보통 흐름(normal flow)으로부터 빠져 텍스트 및 인라인(inline) 요소가 그 주위를 감싸는 자기 컨테이너의 좌우측을 따라 배치되어야 함을 지정합니다. 부동(floating) 요소 는 float 의 계산값(computed value)이 none이 아닌 요소입니다.</p>
  </div>

  <h2>대부분의 요소에 display 값을 block으로 변경함.</h2>
  <div class="container" style="width:400px;">
    <span style="float:left;">inline1</span>
    <span>inline2</span>
  </div>
</body>
</html>
```

https://www.w3schools.com/cssref/pr_class_float.asp

https://www.w3schools.com/cssref/pr_class_float.asp

4) 속성 - clear

- clear 속성 : 요소를 floating 된 요소의 영향에서 벗어나게 하는 속성입니다.

기본값 : none

clear: none | left | right | both | initial | inherit;
 <속성값>
none : 양쪽으로 floating 요소를 허용(기본값)
left : 왼쪽으로 floating 요소를 허용하지 않음
right : 오른쪽으로 floating 요소를 허용하지 않음
both : 양쪽으로 floating 요소를 허용하지 않음
 block-level 요소만 적용 가능
<코드실습>

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>clear</title>
  <style>
    .container {

      border: 1px dashed #aaa;
    }

    div,
    span {
      padding: 20px;
      border: 1px solid red;
    }
  </style>
</head>
<body>
  <h2>clear: none</h2>

  <div class="container">
    <div style="float:left">float:left;</div>
    <span style="display:block;clear:none;">내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로</span>
  </div>

  <h2>clear: left</h2>
  <div class="container">
    <div style="float:left">float:left;</div>
    <span style="display:block;clear:left;">내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로</span>
  </div>
```

```
  <h2>clear: right</h2>
  <div class="container">
    <div style="float:right">float:right;</div>
    <span style="display:inline-block;clear:right;">내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로</span>
  </div>

  <h2>clear: both</h2>
  <div class="container">
    <div style="float:left">float:left;</div>
    <div style="float:right">float:right;</div>
    <span style="display:block;clear:both;">내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로</span>
  </div>
</body>
</html>
```

https://www.w3schools.com/cssref/pr_class_clear.asp
https://developer.mozilla.org/en-US/docs/Web/CSS/clear

5) 속성 - position

- position 속성 : 요소의 위치를 정하는 방법을 지정하는 속성입니다.

기본값 : static

```
position: static | absolute | fixed | relative | sticky | initial | inherit;
```

<속성값>

static : Normal-flow 에 따라 배치되며 offset 값이 적용되지 않는다. (기본값)

absolute : 부모 요소의 위치를 기준으로 offset 에 따라 배치된다. 부모가 position 값(static 제외)을 가지면 offset 값의 시작점이 된다. * 부모의 position 값이 static 인 경우 조상의 position 값이 static이 아닐 때까지 거슬러 올라가 기준으로 삼습니다. Normal-flow의 흐름에서 벗어난다.

fixed : 뷰포트(브라우저의 창)를 기준으로 offset 에 따라 배치된다. 즉, 화면 스크롤에 관계없이 항상 화면의 정해진 위치에 정보가 나타난다. 부모의 위치에 영향을 받지 않는다.

relative : 자신이 원래 있어야 할 위치를 기준으로 offset 에 따라 배치된다. 부모의 position 속성에 영향을 받지 않는다. Normal -flow의 흐름에 따른다. 주변 요소에 영향을 주지 않으면서 offset 값으로 이동한다.

Normal-flow란? 일반적인 상황에서 각의 요소들의 성질에 따라 배치 되는 순서(흐름)를 뜻합니다. 예를 들면, block 레벨 요소들은 상하로 배치되고, inline 레벨 요소들은 좌우로 배치되는 것을 말합니다.

- offset(top/left/bottom/right)

```
top|bottom|left|right: auto|length|initial|inherit;
top: 50%;
left: 10px;
bottom: -10px;
right: auto;
```

offset의 % 단위 사용 : 이전에 padding과 margin에서 % 값을 적용할 때, 상하좌우 방향에 관계없이

가로 사이즈를 기준으로 % 값을 계산된다고 배웠습니다. 그러나 offset은 top, bottom (상하)는 기준이 되는 요소의 height 값 left, right (좌우)는 width값에 대하여 계산 되어집니다.

<코드실습>

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>position</title>
  <style>
    .sibling {
      padding: 5px;
      background-color: #eee;
      text-align: center;
    }

    .parent {
      border: 1px dashed #aaa;
      padding: 10px;
    }

    .child,
    .child_offset {
      width: 60px;
      height: 60px;
      padding: 20px;
      background-color: #dc3636;
      text-align: center;
      color: #fff;
      font-weight: bold;
      border: 1px solid #333;
    }

    .child_offset {
      background-color: #3677dc;
    }

    .static {
      position: static;
    }

    .relative {
      position: relative;
    }
```

```
    .static {
      position: static;
    }

    .absolute {
      position: absolute;
    }

    .fixed {
      position: fixed;
    }
  </style>
</head>
<body>
  <h1>position (Offset 미지정)</h1>
  <h2>static (기본)</h2>
  <div class="parent">
    <div class="sibling">Sibling 1</div>
    <div class="child static">static</div>
    <div class="sibling">Sibling 2</div>
  </div>

  <h2>relative</h2>
  <div class="parent">
    <div class="sibling">Sibling 1</div>
    <div class="child relative">relative</div>
    <div class="sibling">Sibling 2</div>
  </div>

  <h2>absolute</h2>
  <div class="parent">
    <div class="sibling">Sibling 1</div>
    <!-- absolute, fixed는 inline요소일때 display:block으로 변경시킴. inline-block 값을 임의 지정했을 때는 그 값을 유지함. -->
    <span class="child absolute">absolute</span>
    <div class="sibling">Sibling 2</div>
  </div>

  <h2>fixed</h2>
  <div class="parent">
    <div class="sibling">Sibling 1</div>
    <!-- 현재 fixed는 offset값이 없어 뷰포트 밖으로 밀려나오면서 화면 아래로 사라짐 -->
    <div class="child fixed">fixed</div>
    <div class="sibling">Sibling 2</div>
```

```
  </div>


  <h1>position (Offset 지정)</h1>
  <h2>static - </h2>
  <div class="parent">
    <div class="sibling">Sibling 1</div>
    <div class="child_offset static" style="top: 40px; right: 40px;">static <br>top: 40, left: 40</div>
    <div class="sibling">Sibling 2</div>
  </div>

  <h2>relative</h2>
  <div class="parent">
    <div class="sibling">Sibling 1</div>
    <div class="child_offset relative" style="top: 40px; left: 40px;">relative <br>top: 40, left: 40</div>
    <div class="sibling">Sibling 2</div>
  </div>

  <h2>absolute - parent relative</h2>
  <div class="parent relative">
    <div class="sibling">Sibling 1</div>
    <div class="child_offset absolute" style="top: 40px; left: 80px;">absolute<br>top: 40, left: 80</div>
    <div class="sibling">Sibling 2</div>
  </div>

  <h2>fixed</h2>
  <!-- 부모가 fixed여도 별개로 위치함 -->
  <div class="parent">
    <div class="sibling">Sibling 1</div>
    <div class="child_offset fixed" style="top:100px;left:80%;">fixed<br>top:100, left:80%</div>
    <div class="sibling">Sibling 2</div>
  </div>
</body>
</html>
```

https://www.w3schools.com/cssref/pr_class_position.asp
https://developer.mozilla.org/en-US/docs/Web/CSS/position

6) 속성- z-index

- z-index 속성 : 요소가 겹치는 순서(쌓임 순서 또는 stack order)를 지정하는 속성입니다.

기본값 : auto

z-index: auto | number | initial | inherit;

<속성값>
auto : 쌓임 순서를 부모와 동일하게 설정(기본값)
number : 해당 수치로 쌓임 순서를 설정(음수 허용)
z-index: 1;
position 값이 static이 아닌 경우 지정가능
순서 값이 없을 경우 생성순서(코드상 순서)에 따라 쌓임
부모가 z-index 값이 있을 경우 부모 안에서만 의미있음
큰 값이 가장 위쪽(음수 사용 가능)
<코드실습>

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>z-index</title>
  <style>
    .parent {
      z-index: 10;
      position: relative;
      width: 300px;
      height: 50px;
      border: 2px solid #000;
      background-color: #ccc;
    }

    .child {
      z-index: 10;
      position: absolute;
      top: 10px;
      right: 10px;
      width: 100px;
      height: 100px;
      background-color: pink;
      border: 2px solid red;
    }
  </style>
</head>
<body>
  <div class="wrap">

    <h1 class="practive_title">z-index 속성</h1>
    <div class="z_area">
      <div class="parent" style="z-index:11">
        position: relative;
        <div class="child" style="right:20px;z-index:1000;">
```

```
            position: absolute;
          </div>
        </div>
        <div class="parent" style="">
          position: relative;
          <div class="child">
            position: absolute;
          </div>
        </div>
      </div>
</div>
</body>
</html>
```

https://codepen.io/yongwon/pen/dXwyQq

https://www.w3schools.com/cssref/playit.asp?filename=playcss_z-index&preval=2

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index

7) HTML/CSS 유효성 검사

https://validator.w3.org/

8. 미디어쿼리

1) 미디어쿼리 소개 : 미디어쿼리(Media Queries)는 각 미디어 매체에 따라 다른 스타일(css style)을 적용할 수 있게 만드는 것입니다. 미디어 매체는 모니터와 같은 스크린 매체, 프린트, 스크린 리더기와 같은 것들을 이야기 합니다. 미디어쿼리는 동일한 웹 페이지를 다양한 환경의 사용자들에게 최적화된 경험을 제공할 수 있게 해줍니다. CSS2의 미디어 타입(Media Types)을 확장해서 만들어졌습니다. 미디어타입은 이론적으로는 훌륭했지만, 결과적으로 제대로 활용되지 못 했습니다. 당시에는 미디어 타입을 제대로 지원하는 기기가 없었기 때문입니다. 미디어쿼리가 등장하기 이전에는 제대로 된 반응형 웹 사이트를 제작할 수는 없었습니다. 하지만 당시에는 사용자들의 환경이 아주 제한적이었기 때문에 제작자 입장에서는 대중적인 미디어 범위에서만 잘 보이도록 사이트를 제작하면 반응형이 아니더라도 충분했습니다. 하지만 웹이 급격이 발전하면서 대응해야 하는 미디어의 폭이 상당히 늘어났습니다. 이런 필요성에 따라 W3C는 CSS2의 미디어 타입을 확장하여, CSS3 미디어쿼리를 발표합니다. 이 미디어쿼리로 인해 웹 사이트를 제작함에 있어 이전의 정적인 고정 레이아웃 웹 사이트에서 동적으로 반응하는 반응형 웹 사이트로 패러다임이 새롭게 변화하였습니다.

https://www.w3.org/TR/1998/WD-css2-19980128/media.html
https://www.w3.org/TR/css3-mediaqueries/

2) 미디어 타입 & 미디어 특성

- @media(at media) : 미디어쿼리는 CSS2 Media Types을 확장했기 때문에 선언 방법은 동일합니다.

```
@media mediaqueries { /* style rules */ }
```

@media 로 시작하며, 이 키워드는 이제부터 미디어쿼리를 시작한다 라는 뜻입니다. 그 뒤에 미디어쿼리 구문(위 코드의 mediaqueries) 이 나오고 이어서 중괄호({ })를 이용해서 스타일 규칙이 들어갑니다. 미디어쿼리 구문은 논리적으로 평가되며 참이면 뒤에 나오는 스타일 규칙이 적용되고, 거짓이면 무시됩니다. 미디어쿼리 구문은 미디어 타입(Media Types)과 미디어 특성(Media Features)으로 이루어져 있습니다.

- 미디어 타입 : all, braille, embossed, handheld, print, projection, screen, speech, tty, tv

우리가 알아야 할 타입은 all, print, screen 정도입니다. 그 중에서도 screen이 거의 대부분입니다. 화면을 출력하는 디스플레이가 있는 미디어들은 전부 screen에 속하기 때문에 현실적으로 고려해야하는 미디어들은 전부 여기에 해당이 됩니다. print 타입도 간혹 사용이 됩니다. 실습할 때 다룹니다. all 타입은 모든 미디어에 적용되는 타입입니다. 미디어를 구분하는 용도가 아니기 때문에 유용하게 사용되지는 않습니다.

- 미디어 특성 : width, height, device-width, device-height, orientation, aspect-ratio, device-aspect-ratio, color, color-index, monochrome, resolution, scan, grid

미디어 특성 역시 우리가 알아야 할 특성은 width와 orientation 정도입니다. width는 뷰포트의 너비, 즉, 브라우저 창의 너비를 말합니다.(스크린의 크기 x) orientation은 미디어가 세로모드인지 가로모드인지를 구분합니다. 미디어쿼리에서는 이 구분을 width와 height 특성의 값을 비교해서 height가 width보다 같거나 크면 세로모드 반대인 경우에는 가로모드라고 해석합니다. 세로모드에서는 portrait, 가로모드에서는 landscape 키워드와 매칭이 됩니다.

- 미디어쿼리 level 4 : 우리 수업의 미디어 타입과 미디어 특성은 CSS3 미디어쿼리 표준 명세를 기준으로 작성되었습니다. 현재, 미디어쿼리 level 4가 CR(유력 후보안) 단계입니다. 해당 문서에서 미디어 타입 대부분과 미디어 특성 중 일부 속성이 폐기 예정입니다.

https://www.w3.org/TR/css3-mediaqueries/#media1
https://www.w3.org/TR/mediaqueries-4/#media-types
https://www.w3.org/TR/mediaqueries-4/#mf-deprecated

3) Syntax

- 미디어쿼리 Syntax

```
media_query_list
 : S* [media_query [ ',' S* media_query ]* ]?
 ;
media_query
 : [ONLY | NOT]? S* media_type S* [ AND S* expression ]*
 | expression [ AND S* expression ]*
 ;
expression
 : '(' S* media_feature S* [ ':' S* expr ]? ')' S*
 ;
```

위 코드는 CSS3 미디어쿼리 표준 명세에 나와 있는 Syntax 부분입니다. 우리가 알아야 할 부분만 적어 놓았습니다. 참고로 Syntax는 전부 이해할 필요는 없지만, 일부 기호는 알아두면 좋습니다.

[a] : a가 나올 수도 있고 나오지 않을 수도 있습니다.

a | b : a 또는 b 둘 중에 하나를 선택합니다.

"|" : 파이프 라인 기호로 키보드의 역슬래시(\) 키를 Shift 키를 누른 채로 누르면 나옵니다.

a? : a가 0번 나오거나 1번만 나올 수 있음

a* : a가 0번 나오거나 그 이상 계속 나올 수 있음

media_type : all, screen, print 등 명세에 정의된 미디어 타입

media_feature : width, orientation 등 명세에 정의된 미디어 특성

media_query_list : 여러 개의 미디어쿼리로 이루어진 리스트로 작성 가능하며, 쉼표를 이용해서 구분합니다.

media_query : A 형태 - 미디어 타입에 and 키워드를 이용해서 미디어 표현식을 붙일 수 있습니다.
미디어 타입 앞에는 only 또는 not 키워드가 올 수 있습니다.
미디어 표현식은 생략 가능하기 때문에 미디어 타입 단독으로 사용될 수 있습니다.

: B 형태 - 미디어 타입 없이 미디어 표현식이 바로 나올 수 있습니다.(미디어 타입이 명시되지 않으면 all로 간주합니다.) 미디어 표현식은 and 키워드를 이용해서 계속해서 나올 수 있습니다.

expression : 미디어 표현식은 괄호로 감싸야 하며, 특성 이름과 해당하는 값으로 이루어집니다. 이름과 값은 : 기호로 연결합니다. 또, 값이 없이 특성 이름만으로도 작성할 수 있다.

- min-/max- 접두사 : 미디어 특성은 이름 앞에 min- 또는 max- 접두사를 붙일 수 있습니다. 실제로 반응형 사이트를 제작할 때는 보통 접두사를 붙여서 사용합니다. 접두사를 붙이지 않고 사용하는 경우 대부분 효율적이지 못하기 때문입니다. 예를 들어 대부분의 반응형 사이트는 width 특성을 이용하는데, 접두사 없이 width: 00px 이라고 하게 선언하면 정확히 뷰포트의 크기가 00px 에서만 적용되기 때문에, 다양한 기기들을 대응하기 힘듭니다. 그래서 접두사를 사용하여 범위를 지정하게 되면 훨씬 간결하게 반응형 사이트를 제작할 수 있습니다.

- 예제 코드 : 위에서 정의한 Syntax 따라 유효한 미디어쿼리 예제 코드를 살펴보고 어떻게 해석이 되는지

@media (min-width: 768px) and (max-width: 1024px) { ... } : and는 연결된 모든 표현식이 참이면 적용됩니다.(and 키워드는 연결된 부분이 모두 참이어야 적용이 됩니다.)

@media (color-index) : 미디어 장치가 color-index를 지원하면 적용됩니다.

@media screen and (min-width: 768px), screen and (orientation: portrait), ... : 쉼표로 연결된 미디어쿼리 중 하나라도 참이면 적용됩니다.(and 키워드와 반대라고 생각하면 됩니다.)

@media not screen and (min-width: 768px)

: not 키워드는 하나의 media_query 전체를 부정합니다.

: (not screen) and (min-width: 768px) 잘못된 해석!
: not (screen and (min-width: 768px)) 올바른 해석!
: @media not screen and (min-width: 768px), print
첫 번째 미디어쿼리에만 not 키워드가 적용되며, 두 번째 미디어쿼리(print)에는 영향이 없습니다.
- 미디어쿼리 선언 방법
@media screen and (color) : CSS 파일 내부에 또는 <style> 태그 내부에 사용가능 합니다. 대부분의 경우 이렇게 사용합니다.
<link rel="stylesheet" media="screen and (color)" href="example.css"> : <link> 태그의 media 속성에 미디어쿼리를 선언합니다. 미디어쿼리가 참이면 뒤에 css 파일 규칙이 적용됩니다.
@import url(example.css) screen and (color); : CSS 파일 내부에 또는 <style> 태그 내부에 사용가능 합니다. @import문 뒤에 미디어쿼리를 선언하면 됩니다.

4) 실습

- 실습 1. 디스플레이 크기에 따른 body요소의 background-color 변경 : 0~767px 이면 : gold, 768px~1024px 이면 : lightblue, 1025px~ 이면 : lightpink

```
@media (max-width: 767px) {
        body { background-color: gold; }
}
@media (min-width: 768px) and (max-width: 1024px) {
        body { background-color: lightblue; }
}
@media (min-width: 1025px) {
        body { background-color: lightpink; }
}
```

위 코드의 경우, 3개의 조건을 만족하기 위해 3개의 미디어쿼리를 작성했습니다. 위 코드도 우리가 의도한 대로 정상적으로 잘 동작하지만, 1개의 조건을 기본 body 요소의 배경 색상으로 지정해놓게되면 다른 2개의 미디어쿼리만 이용해도 가능합니다. 어떤 조건을 기본으로 정할지는 작성하는 사이트가 모바일 사이트인지, 데스크탑 사이트인지를 먼저 구분해야 합니다. 모바일 중심의 사이트라면(모바일 first) 모바일에 해당하는 조건의 배경 색상을 기본으로 선언하면 되고, 데스크탑 중심의 사이트라면(데스크탑 first) 데스크탑에 해당하는 조건의 배경 색상을 기본으로 선언해놓으면 됩니다. 만들어진 미디어쿼리를 수정해서, 모바일 first인 경우 또 데스크탑 first인 경우대로 한 번씩 작성을 해보겠습니다.

mobile first

```
/* mobile first*/
body { background-color: gold; }
@media (min-width: 768px) and (max-width: 1024px) {
        body { background-color: lightblue; }
}
@media (min-width: 1025px) {
        body { background-color: lightpink; }
}
```

desktop first

```
/* desktop first*/
body { background-color: lightpink; }
@media (min-width: 768px) and (max-width: 1024px) {
```

```
        body { background-color: lightblue; }
}
@media (max-width: 767px) {
        body { background-color: gold; }
}
```

- 실습2. 웹 페이지를 인쇄하는 경우의 스타일 추가 : 앵커 요소의 url 출력, 앵커 요소의 밑줄 제거

미디어 타입 print를 이용하면, 인쇄될 경우에 적용되는 스타일을 추가할 수 있습니다. 먼저 앵커 요소의 url 을 텍스트 뒤에 붙여서 나타나게 하고, 링크임을 표시해주는 밑줄도 제거를 하겠습니다. 웹 페이지를 인쇄할 경우에는 앵커 요소가 가리키는 url을 문서에 같이 출력해주는게 내용을 이해하는데 훨씬 좋습니다. 실제 인쇄를 하지 않더라도 브라우저에서 제공하는 인쇄 미리보기 기능을 이용하면 화면으로 확인이 가능합니다.

```
@media print {
        a { text-decoration: none; }
        a:after { display: inline; content: '(' attr(href) ')'; }
}
```

<코드실습>

```
<!DOCTYPE html>
<html lang="ko">
<head>
        <meta charset="UTF-8">
        <title>Media Queries</title>
        <style>
                /* 1 */
                @media (max-width: 767px) {
                        body { background-color: gold; }
                }
                @media (min-width: 768px) and (max-width: 1024px) {
                        body { background-color: lightblue; }
                }
                @media (min-width: 1025px) {
                        body { background-color: lightpink; }
                }

                /* 2 mobile first*/
                body { background-color: gold; }
                @media (min-width: 768px) {
                        body { background-color: lightblue; }
                }
                @media (min-width: 1025px) {
                        body { background-color: lightpink; }
                }

                /* 3 desktop first*/
                body { background-color: lightpink; }
```

```
                @media (max-width: 1024px) {
                        body { background-color: lightblue; }
                }
                @media (max-width: 767px) {
                        body { background-color: gold; }
                }

                /* 4 print */
                @media print {
                        a { text-decoration: none; }
                        a:after { display: inline; content: '(' attr(href) ')'; }
                }
        </style>
</head>
<body>
        <p>W3C는 <a href="https://www.w3.org/TR/css3-mediaqueries/">CSS3 미디어쿼리 문서</a>를 2012년 6월에 표준 권고안으로 제정하였습니다.</p>
        <p>또한, 기존의 미디어쿼리 개선 작업을 진행 중이며, 2017년 9월에 <a href="https://www.w3.org/TR/mediaqueries-4/">미디어쿼리 레벨4</a>를 발표했습니다. 이 문서는 현재 유력 표준 권고안입니다.</p>
</body>
</html>
```

https://www.w3.org/TR/mediaqueries-4/#mq-min-max

+ 추가. Viewport 이해

- 뷰포트 설정 : 뷰포트를 설정하는 태그는 <meta> 태그로 <head> 태그에 위치해야 합니다. name 속성에 "viewport"라고 선언하며 content 속성에 뷰포트를 설정하는 내용이 들어갑니다.

<meta name="viewport" content=" 뷰포트의 설정 값" >

content 설정

width(height) : 뷰포트의 가로(세로) 크기를 지정합니다. px단위의 수치가 들어갈 수 있지만, 대부분 특수 키워드인 "device-width(height)"를 사용합니다.(뷰포트의 크기를 기기의 스크린 width(height) 크기로 설정한다는 의미입니다.)

initial-scale : 페이지가 처음 나타날 때 초기 줌 레벨 값을 설정합니다.(소수값)

user-scalable : 사용자의 확대/축소 기능을 설정할 수 있습니다.

모바일 웹 사이트의 뷰포트 설정

<meta name="viewport" content="width=device-width, initial-scale=1.0">

https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag

https://www.w3schools.com/css/css_rwd_viewport.asp