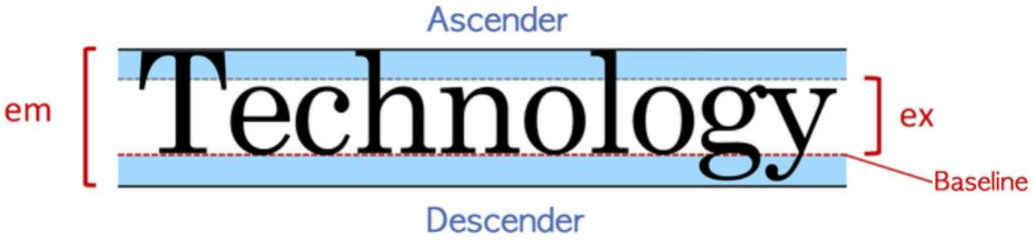
1 CSS-2

6.폰트, 텍스트

1) 속성-typography



모든 폰트는 em박스를 가지고 있고 위 그림과 같은 구조로 이루어져 있습니다.
• em 폰트의 전체 높이를 의미합니다.
• ex (= x-height) 해당 폰트의 영문 소문자 x의 높이를 의미합니다.
• Baseline 소문자 x를 기준으로 하단의 라인을 의미합니다.
• Descender 소문자에서 baseline 아래로 쳐지는 영역을 의미합니다. 서체에 따라 descender의 길이가 다릅니다. (g, j, p, q, y)
• Ascender 소문자 x의 상단 라인 위로 넘어가는 영역을 의미합니다. (b, d, h, l)

2) 속성-font-family

font-family 속성

글꼴을 지정하는 속성입니다.
font-family: family-name | generic-family (| initial | inherit);

- family-name: 사용할 폰트의 이름을 나타내며 ' , ' 로 구분하여 여러 개 선언 할 수 있습니다. 먼저 선언된 순서대로 우선순위가 결정됩니다. 이름 중간에 공백이 있거나, 한글일 경우 홑따옴표로 묶어서 선언합니다.

- generic-family: family-name으로 지정된 글꼴을 사용할 수 없을 경우를 대비해, 브라우저가 대체할 수 있는 폰트가 필요한 경우 선택할 수 있게 해줍니다. font-family 속성의 맨 마지막에 선언해야 하며, 키워드이기 때문에 따옴표 등의 인용부호로 묶지 않는 것이 원칙입니다. 예를 들면 아래와 같이 선언하여 사용할 수 있습니다.

font-family: Helvetica, Dotum, '돋움', Apple SD Gothic Neo, sans-serif;
가장 먼저 Helvetica를 사용하고, 이를 사용할 수 없을 때 Dotum을 사용하는 방식으로 우선순위에 따라 차례대로 적용 됩니다. 만약 "abc 가나다 123" 이라는 글자가 있다면, "abc"와 "123"은 Helvetica로 표현이 되고, "가나다"는 Dotum으로 표현이 됩니다.
"가나다"가 Dotum으로 표현된 이유는 Helvetica는 한글을 지원하는 폰트가 아니기 때문입니다.

그리고 예를 보면 돋움체를 영문으로 한번, 한글로 한번 선언 하였습니다. 왜 이렇게 선언 하였을까요?

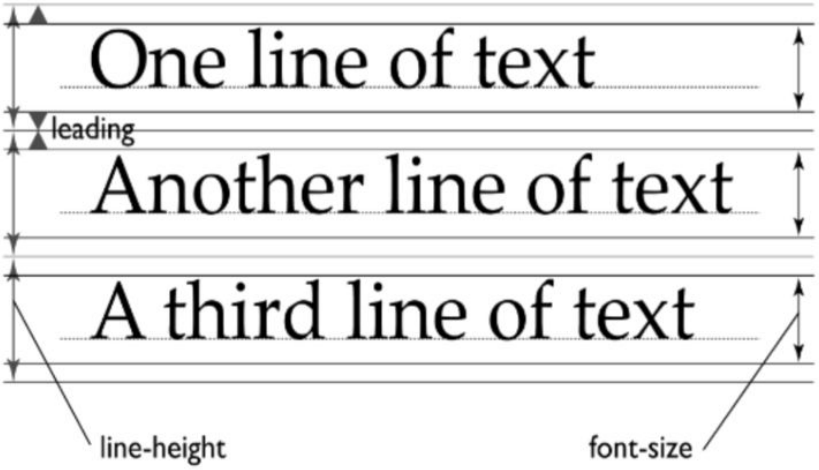
한글을 지원하지 않는 디바이스일 경우 해당 한글 폰트를 불러올 수 없으므로 영문명으로도 선언해 주어야합니다. 마지막에는 반드시 generic-family를 선언 해주어야 합니다. 그 이유는 선언된 모든 서체를 사용할 수 있다는 보장이 없기 때문입니다. 이때 generic-family를 선언해주면, 시스템 폰트 내에서 사용자가 의도한 스타일과 유사한 서체로 적용되기 때문입니다. 또한, 자식 요소에서 font-family를 재선언하면 부모에 generic-family 가 선언되어있어도 다시 선언해주어야 합니다.

Generic-Family에는 대표적인 서체로 serif, sans-serif가 있습니다. serif는 삐침이라는 뜻이고, sans는 프랑스어로 '~이 없이'라는 의미가 있습니다. serif는 글자 획에 삐침이 있는 폰트로 대표적으로 명조체가 있으며, sans-serif는 획에 삐침이 없는 폰트로 대표적으로 돋움체가 있습니다.

3) 속성-line-height

line-height 속성

line-height는 텍스트 라인의 높이를 의미하는 것으로 주로 행간을 제어할 때 사용됩니다.
line-height: normal | number | length | initial | inherit;



행간을 제어할 때 사용하는 속성이라 해서 줄 간격으로 생각해 오해하기 쉬울 수가 있습니다.
줄 바꿈이 되었을 때, 윗줄의 텍스트 하단과 아랫줄의 텍스트 상단까지의 간격이라고 생각할 수도 있지만,
line-height로 제어되는 부분을 line-box라고도 하며 이는 타이포그래피 구조에서 배웠던 [em 박스] + [상하단의 여백]까지를 의미합니다.
보통 폰트 사이즈보다 2~4px정도 더한 값을 사용합니다.
〈속성 값〉

normal 기본값으로 브라우저의 기본 속성을 따릅니다.
폰트에 따라 브라우저에 따라 다르지만 보통 1.2 정도로 할당되어 있습니다.

number font-size를 기준으로 설정한 숫자만큼 배율로 적용합니다.

length px, em 등 고정 수치로 할당할 수 있습니다.

% font-size를 기준으로 설정한 퍼센트만큼 배율로 적용합니다.

주의할 점은, line-height의 값으로 number를 선언할 때와 %로 선언할 때의 차이입니다.
두 값 모두 font-size를 기준으로 동작하기 때문에 1이나 100%를 같은 것이라고 오해할 수 있습니다.
하지만 두 값은 큰 차이가 있습니다. 바로 line-height의 값이 자식 요소로 상속되었을 때의 계산 방식입니다.

- number 부모 요소의 숫자 값이 그대로 상속됩니다. 즉, 자식 요소에서도 또 한 번 자식 요소의 font-size를 기준으로 계산해야 합니다.

- **%** 부모 요소에서 %값이 그대로 상속되는 것이 아니고, %에 의해 이미 계산된 px값이 그대로 상속됩니다.

아래 코드를 예시로 보자면,
```
body { font-size: 20px; line-height: 2; }      /* line-height = 40px; */
body { font-size: 20px; line-height: 200%; }   /* line-height = 40px; */
```
두 경우 모두 <body>에 똑같이 line-height: 40px이 적용됩니다.

하지만 자식 요소로 <p>가 있다고 생각을 하면 얘기가 달라집니다.
```
body { font-size: 20px; line-height: 2; }   /* line-height = 40px; */
p { font-size: 10px; }                      /* line-height = 20px; */

body { font-size: 20px; line-height: 200%; }   /* line-height = 40px; */
p { font-size: 10px; }                         /* line-height = 40px; */
```
이처럼 계산된 값이 아닌 숫자 값을 상속한다는 사실 때문에,
숫자 값을 사용하면 부모 엘리먼트에서 계산된 값 대신 비율을 그대로 상속받을 수 있으므로,
가능하면 단위가 없는 값을 사용하는 것이 좋습니다.
기본 값 : normal

(1.5em == 24px)
Quiz. 아래 코드를 보고 .child의 line-height 값으로 알맞은 것을 고르시오
```
<div class="wrap">
<div class="parent">
<div class="child">line-height 값 계산</div>
</div>
</div>
.wrap { font-size: 20px; line-height: 2; }                /* line-height = 40px; */
.parent { font-size: 1.5em; line-height: 200%; }          /* line-height = 20px*1.5배*2배 = 60px; */
.child { font-size: 15px; }        그대로 상속
```

따라서 60px

4) 속성-font-size

font-size 속성
글꼴의 크기를 지정하는 속성입니다.
기본 값 : medium
< 속성 값 >
keyword (추천X) medium(기본 값), xx-small, x-small, small, large, x-large, xx-large, smaller, larger
length px, em 등 고정 수치로 지정합니다. - 추천!
% 부모 요소의 font-size 기준의 퍼센트로 지정합니다.

font-size: keyword | length | initial | inherit ;

- **absolute size (keyword)** 기본 값인 medium에 대한 상대적인 크기로, 브라우저마다 사이즈가 다르게 정의되어있습니다.

- **relative size (keyword)** 부모 요소의 font-size 크기에 대해 상대적입니다. smaller는 0.8배, larger는 1.2배입니다.

- **length** px, em, rem 등의 단위를 이용하여 고정된 크기를 지정할 수 있습니다. - em : 부모 요소의 font-size에 em 값을 곱한 크기 - rem : 루트의 font-size에 rem 값을 곱한 크기

- **percent (%)** 부모 요소의 font-size를 기준으로 백분율 계산된 값을 지정할 수 있습니다.

- **viewport units** vw, vh 단위로 뷰포트를 기준으로 하여, 유동적인 font-size를 지정할 수 있습니다. vw는 뷰포트 width의 1%, vh는 뷰포트 height의 1% 값을 가집니다.

5) 속성-font-weight

font-weight 속성
글꼴의 굵기를 지정하는 속성입니다.

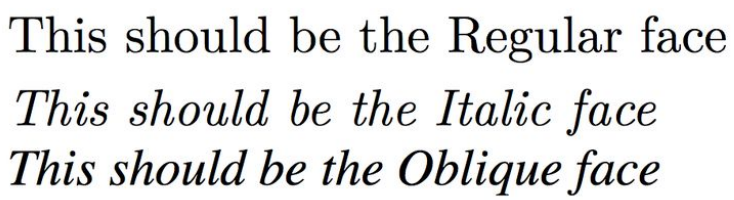
기본 값 : normal
font-weight: normal | bold | bolder | lighter | number | initial | inherit ;

<속성 값>
normal 기본 값 (400)
bold 굵게 표현(700)
bolder 부모 요소 보다 두껍게 표현
lighter 부모 요소 보다 얇게 표현
number 100, 200, 300, 400, 500, 600, 700, 800, 900 (클수록 더 두껍게 표현)
실무에서는 normal과 bold를 많이 사용하고, 부모 요소에 영향이 있는 bolder와 lighter는 사용을 될 수 있으면 지양하는 편입니다.
물론 상속 관계에서 바뀌어야 하는 스펙이라면 당연히 유용하게 사용될 수 있지만, 그 외의 경우에는 사용에 있어 신중해야 합니다.
font-weight는 normal, bold와 같은 키워드 외에 숫자로도 그 굵기를 표현할 수 있습니다.
100~900까지 100단위로 값을 지정하여 사용할 수 있고 숫자가 커질수록 더욱 굵게 표현됩니다.
기본적으로 400은 normal과 같고, 700은 bold와 같습니다
그러나 수치를 이용한 font-weight는 폰트 자체에서 지원을 해야 표현할 수 있습니다.
폰트에 따라 font-weight를 적용해도 굵기에 변화가 없을 수도 있으며,

normal과 bold만 지원하는 폰트일 경우에는 100~500까지는 normal로, 600~900까지는 bold로 표현됩니다.
폰트가 점차 다양해지면서 굵기 자체를 폰트 family-name으로 가지고 있는 경우도 있습니다.
또한, 디바이스별로 조금 다르게 표현될 수도 있습니다.
font-weight와 font-family, font-size는 서로 밀접하게 연관되어있습니다.

6) 속성-font-style

font-style 속성
글꼴의 스타일을 지정하는 속성입니다.
기본 값 : normal
font-style: normal | italic | oblique | initial | inherit;



〈속성 값〉

normal	font-family 내에 분류된 기본 값
italic	italic 스타일로 표현합니다.
oblique	oblique 스타일로 표현합니다.

oblique 텍스트의 기울기에 대한 각도를 추가로 지정할 수 있습니다. 브라우져 별 지원율이 다름 아직은 많이 X

font-weight oblique 〈각도〉;
유효한 값은 -90 ~ 90도이며, 따로 각도를 지정하지 않으면 14도가 사용됩니다. 양수 값은 글의 끝 부분 쪽으로 기울어지며, 음수값은 시작 부분 쪽으로 기울어집니다. 그러나 아직 초안 단계로 CSS Fonts Module Level 4를 지원하는 브라우저에서만 사용 가능합니다.

7) 속성-font-variant

font-variant 속성
글꼴의 형태를 변형하는 속성으로 소문자를 작은 대문자로 변환할 수 있습니다.
기본 값 : normal
font-variant: normal | small-caps | initial | inherit ;

〈속성 값〉

normal	기본 값
small-caps	소문자를 작은 대문자로 변형합니다.

8) 속성-font

font 관련 속성
font-style, font-variant, font-weight, font-size/line-height, font-family 속성들을 한 번에 선언할 수 있는 축약형 속성입니다. 이 폰트 속성은 실무에서 지양하는 속성 !! 그렇지만 누군가 썼을 때 읽을 수는 있어야…!

〈속성 값〉

font: font-style font-variant font-weight font-size/line-height font-family | initial | inherit;

font-style	font-style 지정, 기본 값은 normal
font-variant	font-variant 지정, 기본 값은 normal
font-weight	font-weight 지정, 기본 값은 normal
font-size/line-height	font-size/line-height 지정, 기본 값은 normal
font-family	font-family 지정

```
/* style | size | family */
font: oblique 2em "돋움", dotum, sans-serif;

/* style | variant | weight | size/line-height | family */
font: oblique small-caps bold 16px/1.5 '돋움';

/* The font used in system dialogs */
font: message-box;
font: icon;
```

축약형을 선언할 때는 아래 사항들을 유의해야 합니다.

- font-size와 font-family는 반드시 선언해야 하는 필수 속성입니다.
- 빠진 속성이 있다면 기본 값으로 지정됩니다.
- 각 속성의 선언 순서를 지켜야 합니다.

9) 속성-webfont

@font-face
웹에 있는 글꼴을 사용자의 로컬 환경(컴퓨터)으로 다운로드하여 적용하는 속성입니다.
기본 값 : 없음

```
@font-face {
  font-properties
}
```

〈속성 값〉

font-family(필수)	글꼴의 이름을 지정
src(필수)	다운로드 받을 글꼴의 경로(URL)
font-style(옵션)	글꼴의 스타일 지정, 기본 값은 normal
font-weight(옵션)	글꼴의 굵기 지정, 기본 값은 normal

사용 예시는 다음과 같습니다.

```
@font-face {
  font-family: webNanumGothic; /* 사용자 지정 웹 폰트명 */
  src: url(NanumGothic.eot); /* 적용 될 웹 폰트의 경로 */
  font-weight: bold; /* 필요에 따라 지정 */
  font-style: italic; /* 필요에 따라 지정 */
}

body {
  font-family: webNanumGothic;
}
```

웹폰트는 선언 시 필요에 따라 굵기나 스타일 등을 같이 지정해서 사용할 수 있습니다
참고링크 : https://wit.nts-corp.com/2017/02/13/4258

10) 속성-vertical-align

vertical-align 속성

요소의 수직 정렬을 지정하는 속성입니다.
기본 값 : baseline

```
vertical-align: keyword | length | percent | initial | inherit ;
```

〈속성 값〉

length	요소를 지정한 길이만큼 올리거나 내림. 음수 허용

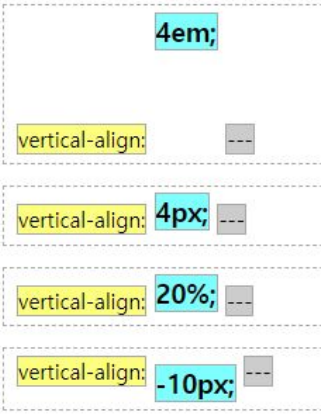
% 요소를 line-height를 기준으로 올리거나 내림. 음수 허용

keyword baseline(기본 값), sub, super, top, text-top, middle, bottom, text-bottom
부모를 기준으로?

vertical-align은 기본 값이 baseline입니다.
이전에 타이포그래피 구조 강의에서 설명했듯이 baseline은 소문자 x를 기준으로 하단 라인을 의미합니다.

- **keyword** sub : 부모 아래 첨자 기준으로 정렬 super : 부모 위 첨자 기준으로 정렬 text-top : 부모 텍스트의 맨 위(Ascender 제외) text-bottom : 부모의 텍스트의 맨 아래(Descender 제외) middle : 부모의 중앙에 위치 top : 부모의 맨 위 위치 bottom : 부모의 맨 아래 위치
- **length** px값 사용 시 baseline을 기준으로 이동하며, 음수 값도 사용 가능합니다.
- **percent (%)** line-height를 기준으로 내에서 이동하며 음수 값 사용 가능합니다.

인라인 레벨에서만 선언할 수 있다! -> 인라인 요소 또는 테이블 셀 상자의 수직 정렬을 지정할 때 사용합니다.



11) 속성-text-align

text-align 속성
텍스트의 정렬을 지정하는 속성입니다.
기본 값 : left (Right to Left 언어일 경우는 right)
text-align: left | right | center | justify | initial | inherit ;

기본 값은 left이지만 경우에 따라 다릅니다.
문서의 방향이 LTR(Left To Right) 왼쪽에서 오른쪽 방향인 언어일 경우 left가 기본값이고,
RTL(Right To Left) 로 오른쪽에서 왼쪽으로 읽힐 경우 right가 기본값이 됩니다.

〈속성 값〉
left 텍스트를 왼쪽으로 정렬
right 텍스트를 오른쪽으로 정렬

center 텍스트를 중앙으로 정렬
justify 텍스트를 라인 양쪽 끝으로 붙여서 정렬. (마지막 라인은 정렬하지 않음)

블럭 요소 안에서 선언, 인라인 요소들이 각각 정렬되는 것이라고 생각하면 됩니다.

text-align과 display의 관계
- text-align은 inline-level에 적용
- text-align은 block-level에 적용할 수 없음

그렇다면 block 요소를 가운데 정렬 하고자 한다면 어떻게 해야 할까요?
박스모델 챕터에서 다룬 margin의 auto 값을 이용해서 하시면 됩니다.

- 가운데 정렬 인라인 요소 : text-align (center)

- 블럭 요소 : margin (auto)

요소의 레벨에 따라 정렬하는 방식의 차이를 바로 알고 있으시기 바랍니다.

12) 속성-text-indent

text-indent 속성
텍스트의 들여쓰기를 지정하는 속성입니다.
기본 값 : 0

text-indent: length | initial | inherit;
〈속성 값〉
length px, em 등 고정 수치로 지정. 음수 허용
% 부모 요소의 width를 기준으로 퍼센트로 지정

- **length** 문단의 첫 줄에 대한 들여쓰기를 수행합니다. 음수 값을 사용할 수 있으며, 음수 값 사용 시 왼쪽으로 이동합니다.

- **percent (%)** 텍스트를 포함하는 컨테이너 블록의 width(부모의 width)를 기준으로 변환된 백분율 값으로 들여쓰기합니다.

13) 속성-text-decoration

text-decoration 속성
텍스트의 장식을 지정하는 속성입니다. 아래 속성들의 단축 속성으로, 기본 값은 차례대로 아래 3가지 속성의 값입니다.
기본 값 : none currentColor solid

text-decoration: text-decoration-line text-decoration-color text-decoration-style | initial | inherit;

- **text-decoration-line** 텍스트 꾸밈의 종류를 지정하는 속성입니다.

- 기본 값 : none

〈속성 값〉
none 텍스트 꾸밈을 생성하지 않음 (기본값)
underline 밑줄로 꾸밈을 설정
overline 윗줄로 꾸밈을 설정
line-through 중간을 지나는 줄로 꾸밈을 설정

<p style="text-decoration: overline;">

- text-decoration-color
- 텍스트 꾸밈의 색상을 지정하는 속성입니다.
- 기본 값 : currentColor
- 색상 값을 사용하여 원하는 색상을 지정할 수 있습니다.

- text-decoration-style 꾸밈에 사용되는 선의 스타일을 지정하는 속성입니다.
- 기본 값 : solid

< 속성 값 >

solid	한줄 스타일 (기본 값)
double	이중선 스타일
dotted	점선 스타일
dashed	파선 스타일
wavy	물결 스타일

+코드 실습

14) 속성-단어 관련 속성

white-space 속성
요소 안에 공백을 어떻게 처리할지 지정하는 속성입니다.
기본 값 : normal
white-space: normal | nowrap | pre | pre-line | pre-wrap | initial | inherit;

< 속성 값 >

normal	공백과 개행을 무시하고, 필요한 경우에 자동 줄바꿈 발생. 기본 값
nowrap	공백과 개행을 무시하고, 자동 줄바꿈이 일어나지 않음.
pre	공백과 개행을 표현하고, 자동 줄바꿈이 일어나지 않음.
pre-line	공백은 무시하고, 개행만 표현. 필요한 경우에 자동 줄바꿈 발생.
pre-wrap	개행은 무시하고, 공백만 표현. 필요한 경우 자동 줄바꿈 발생.

letter-spacing 속성
자간을 지정하는 속성입니다.
기본 값 : normal
letter-spacing: normal | length | initial | inherit;

< 속성 값 >

normal	기본 값
length	길이만큼 자간을 지정. 음수 허용

word-spacing 속성

단어 사이의 간격을 지정하는 속성입니다.
기본 값 : normal
word-spacing: normal|length|initial|inherit;
〈속성 값〉
normal 기본 값
length 길이만큼 단어 사이의 간격을 지정. 음수 허용

word-break 속성
단어가 라인 끝에 나올 경우 어떻게 처리할지(중단점) 지정하는 속성입니다.
기본 값 : normal
word-break: normal | break-all | keep-all | initial | inherit;

〈속성 값〉
normal 기본 값. 중단점은 공백이나 하이픈(-)(CJK는 음절)
break-all 중단점은 음절. 모든 글자가 요소를 벗어나지 않고 개행
keep-all 중단점은 공백이나 하이픈(-)(CJK는 그 외 기호도 포함)

word-wrap 속성
요소를 벗어난 단어의 줄바꿈을 지정하는 속성입니다.
기본 값 : normal
word-wrap: normal|break-word|initial|inherit;
〈속성 값〉
normal 기본 값. 중단점에서 개행
break-word 모든 글자가 요소를 벗어나지 않고 강제로 개행

* word-break와 word-wrap은 같이 썼을 때 어떤 조합에 따라서 굉장히 많이 달라지기에 꼭 테스트 해보면서 공부해보세요~

7. 레이아웃

1) 속성-display

display 속성
요소의 렌더링 박스 유형을 결정하는 속성입니다.
기본 값 : - (요소마다 다름)

display: value;

〈속성 값〉

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 속성 값 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

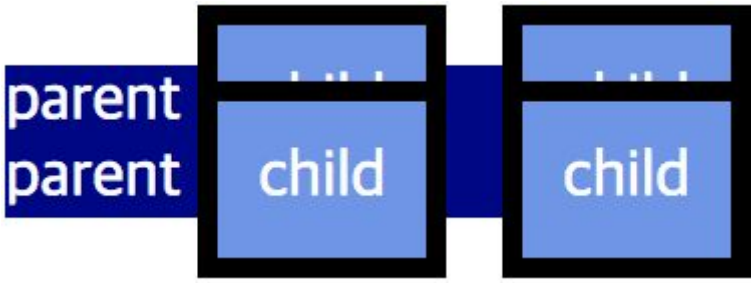
display	width	height	margin	padding	border
block	o	o	o	o	o
inline	X	X	좌/우	o (설명)	o (설명)
inline-block	o	o	o	o	o

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.

parent 는 〈div〉, child 는 〈span〉

하지만 상/하 padding/border는 line-box에는 영향을 주지 못하기 때문에 위와 같이 부모 요소의 박스에 반영되지 않습니다. → 실무에서 잘 쓰지 X 확실하게 적용되는 좌우에만 사용.

어떤 사람: inline 은 높이값을 가지지 않기때문에 inline-block 을 통해서 자기 영역만큼 가로값을 차지하면서 높이값 적용을 해야할 때 저는사용하고 있습니다 :)



parent 는 , child 는

또한 인접한 다른 line-box 에도 반영되지않습니다. 즉 콘텐츠가 겹칠 수 있기 때문에 실무에서는 잘 사용하지 않습니다.
line-box에 대해 궁금하거나 기억이 나지 않으신다면 Chapter06에 line-height 강의를 참고하시면 됩니다.
css로 화면에 보이는 display 속성을 바꿀 수는 있지만 그 태그의 본질을 바꾸는 것은 아니다. 그래서 문법적 오류가 없도록 태그를 작성하셔야 합니다. 의 display를 block으로 바꾼다고 해서 이 <display>가 되는 건 X

정리
블록은 width, height,margin,padding,border값이 적용된다. 인라인 블록도 마찬가지이다.
하지만 인라인은 width,height값이 적용되지 않는다. margin,padding 값도 좌우값만 적용된다.
이처럼 인라인을 인라인 블록으로 만들어서 사용하면 모든것이 적용되는데 다만 글자 사이 4px여백이 발생할수 있다.

2) 속성-visibility

visibility 속성
요소의 화면 표시 여부를 지정하는 속성입니다. 어떻게 숨길지?

기본 값 : visible

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

< 속성 값 >
visible 화면에 표시
hidden 화면에 표시되지 않음(공간은 차지함)
collapse 셀 간의 경계를 무시하고 숨김(테이블 관련 요소에만 적용 가능)

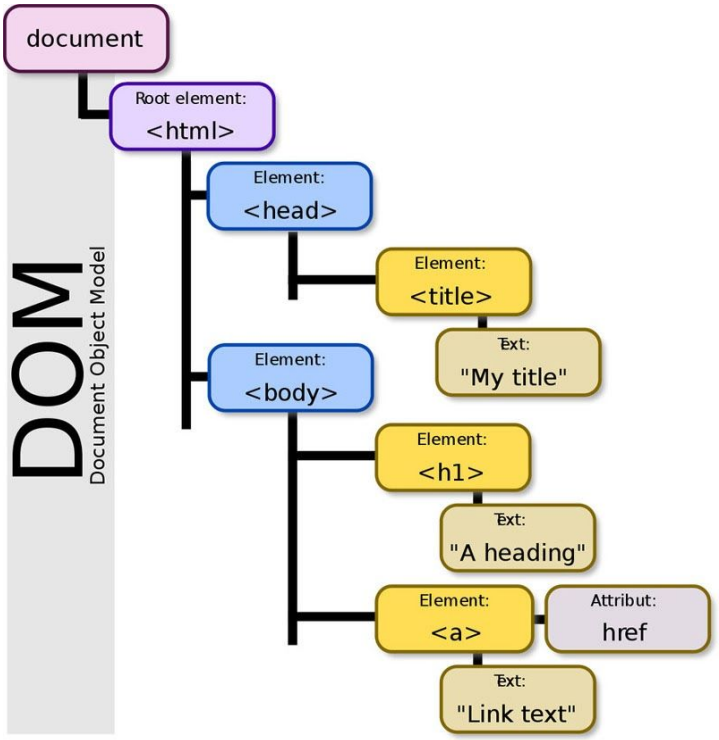
```
visibility: visible;  /* 보임 기본값 */
visibility: hidden;  /* 숨김, 자신의 박스 영역은 유지(margin까지 모두 포함) */
visibility: collapse; /* 셀간의 경계를 무시하고 숨김(박스영역 없음, 테이블의 행과 열 요소에만 지정 가능, 그 외 요소 지정은 hidden과 같음) - 잘 사용 X */
```

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)

스크린 리더기가 hidden 내용은 읽어주지만 none 내용은 읽어주지 않음.

밑은 참고자료



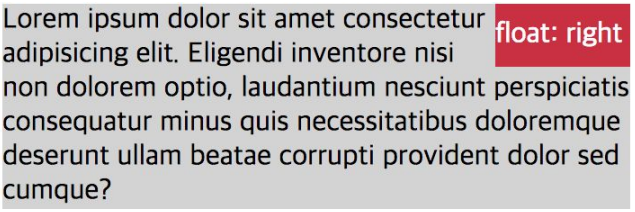
3) 속성-float

float 속성
요소를 float(떠가다 뜨다 요소를 보통의 흐름에서 벗어나게 함) 시킬지 지정하는 속성입니다.
기본 값 : none

float: none | left | right | initial | inherit;
〈속성 값〉
none float 시키지 않음(기본값)
left 좌측으로 float 시킴
right 우측으로 float 시킴

보통의 흐름에서 벗어나서 떴다. 띄워졌다. 하지만 자신의 영역은 차지하고 있다 ! 를 이해하시면 됩니다.

- 요소를 보통의 흐름에서 벗어나 띄어지게 함
- 주변 텍스트나 인라인 요소가 주위를 감싸는 특징이 있음
- float: left만 작성해도 대부분 요소의 display 값을 block으로 변경된다
  (display 값 변경 예외: inline-table, flex 등)



정리
float: 요소를 보통의 흐름에서 벗어나게 함. 붕떠있게 함. 그래도 자신의 영역은 차지 하고 있다. 주변 텍스트나 인라인 요소가 주의를 감싸고 있다. 태그 처럼 인라인 태그를 float:left를 선언하면 자동으로 dispaly:block으로 바뀜으로 불필요하게 다시 선언할 필요가 없다. 인라인테이블이나 플렉스의 경우 디스플레이 값이 자동으로 변경되지 않음을 기억한다.

4) 속성-clear

clear 속성
요소를 floating 된 요소의 영향에서 벗어나 다음 행으로 이동하게 하는 속성입니다.
기본 값 : none
clear: none | left | right | both | initial | inherit;

<속성 값>
none 양쪽으로 floating 요소를 허용(기본값)
left 왼쪽으로 floating 요소를 허용하지 않음
right 오른쪽으로 floating 요소를 허용하지 않음
both 양쪽으로 floating 요소를 허용하지 않음
~쪽에 float된 요소를 모두 clear해주겠다.

- block-level 요소만 적용 가능

display: inline-block 은 inline 요소이므로 clear 제대로 동작 X

```
<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</span>
 </div>
```

우리 집안의 일은 우리 집안에서 해결하고 싶다 - 의미없는 블럭 하나 만들기
근데 clear는 block-level 요소만 적용가능하므로 span에서 block으로 선언!

5) 속성-position

position 속성
요소의 위치를 정하는 방법을 지정하는 속성입니다.
기본 값 : static

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

< 속성 값 >

static Normal-flow 에 따라 배치되며 offset(좌표지정) 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. ---> 다 없으면 viewport 기준, 스크롤 할 땐 body 태그 따라서 움직임*
- Normal-flow의 흐름에서 벗어난다.
- 자동으로 block 화 !!!

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.

좌표지정 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 계산합니다.

6) 속성-z-index

z-index 속성

기본 값 : auto

요소가 겹치는 순서(쌓임 순서 또는 stack order)를 지정하는 속성입니다.

박스가 겹치는 순서를 지정

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

〈속성 값〉

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 부모 간의 싸움!
- 큰 값이 가장 위쪽(음수 사용 가능)

순서값은 auto 0 1 2 3 …

position과 z-index 는 마치 float과 clear의 관계처럼 짝꿍을 이루는 경우가 많아요

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

tool 팁 박스나 도움알 레이어 박스에서 사용

퀴즈

7) HTML/CSS 유효성 검사

https://validator.w3.org/

HTML/CSS 코드를 작성하고 나면 항상 유효성 테스트를 실시해야 합니다.

유효성 검사를 통해 마크업 문법상 에러가 없는지 표준에 맞게 작성되었는지 알 수 있고,

사용성과 접근성, SEO 최적화 개선에도 도움이 됩니다.

아래 링크는 유효성 검사를 할 수 있는 주소입니다.

HTML/CSS이 익숙하지 않을 때는 반드시 코드 작성 후 항상 유효성 검사하는 습관을 들이세요!

8. 미디어쿼리

1) 미디어쿼리 소개

미디어쿼리 소개

미디어 쿼리(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.

미디어쿼리는 CSS2의 미디어 타입(Media Types)을 확장해서 만들어졌습니다.

미디어타입은 이론적으로는 훌륭했지만, 결과적으로 제대로 활용되지 못했습니다.

이유는 당시에는 미디어 타입을 제대로 지원하는 기기가 없었기 때문입니다.

미디어 쿼리가 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 급격히 발전하면서 대응해야 하는 미디어의 폭이 상당히 늘어났습니다.

이런 필요성에 따라 W3C는 CSS2의 미디어 타입을 확장하여, CSS3 미디어쿼리를 발표합니다.

이 미디어 쿼리로 인해 웹 사이트를 제작함에 있어 이전의 정적인 고정 레이아웃 웹 사이트에서

동적으로 반응하는 반응형 웹 사이트로 패러다임이 새롭게 변화하였습니다.

2) 미디어 타입 & 미디어 특성

@media(at media)
미디어 쿼리는 CSS2 Media Types을 확장했기 때문에 선언 방법은 동일합니다.
@media media queries { /* style rules */ }
@media 로 시작하며, 이 키워드는 이제부터 미디어 쿼리를 시작한다 라는 뜻입니다.
그 뒤에 미디어 쿼리 구문(위 코드의 *mediaqueries*) 이 나오고 이어서 중괄호({ })를 이용해서 스타일 규칙이 들어갑니다.
미디어 쿼리 구문은 논리적으로 평가되며 참이면 뒤에 나오는 스타일 규칙이 적용되고, 거짓이면 무시됩니다.
미디어 쿼리 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. (인쇄) 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(가로), 즉 브라우저 창의 너비를 말합니다.(스크린의 크기 x) width: 길이px cm
orientation은 미디어가 세로모드인지 가로모드인지를 구분합니다.
미디어 쿼리에서는 이 구분을 width와 height 특성의 값을 비교해서 height가 width보다 같거나 크면 세로모드 반대인 경우에는 가로모드라고 해석합니다. 세로모드에서는 portrait, 가로모드에서는 landscape 키워드와 매칭이 됩니다. orientation: portrait / orientation: landscape

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

3) 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 미디어 쿼리 표준 명세에 나와 있는 Syntax 부분입니다. 우리가 알아야 할 부분만 적어놓았습니다.
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.
  "|"는 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 …n
- ***media_type*** : all, screen, print 등 명세에 정의된 미디어 타입
- ***media_feature*** : width, orientation 등 명세에 정의된 미디어 특성
- S* 공백과 관련 → 없다고 보기
- ‘,’ ‘)’ 실제로 구문에 이 것이 들어간다

위 Syntax를 요약하면
media_query_list
: 여러개의 미디어 쿼리로 이루어진 리스트로 작성 가능하며, 쉼표를 이용해서 구분합니다. 이렇게 연속적으로 나올 수 있다
media_query
: A 형태 - 미디어 타입에 and 키워드를 이용해서 미디어 표현식을 붙일 수 있습니다.
미디어 타입 앞에만 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 미디어 쿼리 예제 코드를 살펴보고 어떻게 해석이 되는지 같이 봅니다.
@media screen { ... }
: 미디어 타입이 screen이면 적용됩니다.
@media screen and (min-width: 768px) { ... }
: 미디어 타입이 screen이고 width가 768px 이상이면 적용됩니다. 둘 다 참이면 적용됩니다.
@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 미디어 쿼리 중 하나라도 참이면 적용됩니다.(and 키워드와 반대라고 생각하면 됩니다.)
@media not screen and (min-width: 768px)
: not 키워드는 하나의 media_query 전체를 부정합니다. 가장 마지막에 해석됨
: (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 미디어 쿼리에만 not 키워드가 적용되며, 두 번째 미디어 쿼리(print)에는 영향이 없습니다.

미디어 쿼리 선언 방법
미디어 쿼리를 선언하는 3가지 방법에 대해 알아보겠습니다.
참고로 @media를 이용한 방법을 가장 많이 사용하며 나머지 2가지 방법은 거의 쓰이지 않습니다.
@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 미디어 쿼리를 선언합니다. 미디어 쿼리가 참이면 뒤에 css 파일 규칙이 적용됩니다.
@import url(example.css) screen and (color);
: CSS 파일 내부에 또는 <style> 태그 내부에 사용가능 합니다. @import문 뒤에 미디어 쿼리를 선언하면 됩니다. _ 잘 사용이 되지 않습니다.

4) 실습

실습 1. 디스플레이 크기에 따른 body요소의 background-color 변경
세부 조건

- 0~767px 이면 : gold
- 768px~1024px 이면 : lightblue
- 1025px~ 이면 : lightpink

width 미디어 특성을 이용해서 뷰포트 가로 크기에 따라 배경 색상이 변경되는 예제를 만들어봅니다.

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

실습 2. 웹 페이지를 인쇄하는 경우의 스타일 추가
세부 조건

- 앵커 요소의 url 출력
- 앵커 요소의 밑줄 제거

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

근데 왜 난 안먹지…

css3 미디어 쿼리 (강의) https://www.w3.org/TR/css3-mediaqueries/
다음 표준안으로 유력한 미디어 쿼리 레벨 4 https://www.w3.org/TR/mediaqueries-4/