<a> 태그의 속성값 'target'

_self : default값. 현재 창에서 열기

_blank : 새 창으로 열기

_parent : 가장 근접한 부모의 브라우저에서 열기

(창 A에서 창 B를 열었고, 창 B에서 _parent로 창을 열면 창 A에서 열리게 됨)

_top : 여러 중첩된 브라우저가 존재할 때 최상위 브라우저에서 열기

폰트 관련 css 속성

* letter-spacing: 자간 설정. 음수도 가능함 (px, em, rem 등으로 표기)

* border: 1px solid #000;

= border-width: 1px; / border-style: solid; / border-color: #000

풀어서 쓰는 것이 유지보수 및 자바스크립트 접근에 좋음

* text-align: 블록 속성에만 적용이 가능함, span과 같은 인라인 속성에는 적용되지 않음

'em'과 'rem'

'em'을 사용해 폰트 크기 바꾸는 법

em은 기준이 되는 폰트 크기가 있어야 하고, '직계 부모'를 기준으로 작성됨

폰트 크기에 통일성을 부여하므로 현업에선 더러 사용함

Ex) section { font-size: 16px; }

⇨ 1em을 특별한 경우가 아닐 시 16px로 지정함

이후 하위 태그에 1em, 3em 혹은 0.8em 이런식으로 작성 가능

'rem'을 사용해 폰트 크기 바꾸는 법

'rem'의 'r'은 'root'라는 의미로 뿌리, 근본이라는 뜻

따라서 'rem'을 사용할 때는 html의 폰트 크기를 꼭 부여해줘야 함

Ex) html { font-size: 16px; }

⇨ 해당 문서의 1rem을 16px로 지정함

이후 하위 태그에 1em, 3em 혹은 0.8em 이런식으로 작성 가능

text 관련 css 속성

* text-transform: uppercase와 lowercase로 대소문자를 한 번에 변경할 수 있음

<img> 태그의 css 속성

* object-fit의 값

fill: default값, 이미지의 비율을 무시하고 부여된 너비값으로 채움

contain: 이미지의 비율을 유지하면서 '가로 너비'를 채워서 삽입함.

따라서 상하 여백이 생길 수 있음, 이미지의 좌우에 중요한 컨텐츠가 있고 위아래는 중요하지 않거나 의미없는 여백일 때 이 방법이 더 좋을 수 있음

cover: 이미지의 비율을 유지하면서 '세로 너비'를 채워서 삽입함.

따라서 좌우 여백이 생길 수 있음

※ img 태그는 부모에게서 크기를 상속받지 않음

background-image 관련 css 속성

* background-image: url( ); 로 이미지 삽입 가능

* background-position: left, center, top / px / % 으로 표기 가능

* background-size: cover / contain

* background-color: transparent => 배경색이 투명해짐

마진 역상속

부모 블록요소에서 첫번째 자식 블록요소에 마진 탑이 설정될 경우 그 값이 부모 요소로 역상속되는 현상

따라서 패딩값을 주는 것이 좋음

패딩 값으로 높이 주는 법

중요하지 않은 컨텐츠일 경우(존재 자체가 의의인 경우) 따로 높이값을 부여하지 않고 패딩으로 가운데 정렬(세로)이 가능함

같은 칸 안에 두 요소의 글자 크기가 달라서 패딩값 계산이 헷갈릴 경우 align-items를 사용하면 해결됨

스타일 속성 inset

border와 비슷하지만 다른 개념으로, border는 완전한 바깥 영역에 테두리를 만드는 것이지만

inset으로 만든 테두리는 컨텐츠의 안쪽 영역에 생성됨

box-shadow에 값을 부여할 때와 마찬가지로 0px 0px 0px 0px rgba; 의 식으로 방향 별 두께 색상 지정 가능

flex 관련 css 속성

* display: inline-flex; 자기 자신은 인라인 속성이 되고 자식요소를 flex 환경으로 설정

* flex-direction의 값

row: default값, 기본축 가로, 진행방향 좌 > 우

row-reverse: 기본축 가로, 진행방향 우 > 좌

column: 기본축 세로, 진행방향 위 > 아래

column-reverse: 기본축 세로, 진행방향 아래 > 위

※ flex-direction이 column으로 적용되어 있을 경우 gap을 주면 위쪽 여백이 적용됨

* flex-wrap: 줄바꿈 여부에 관여

* justify-content의 값

flex-start: default값, 자식요소를 주축의 시작지점에서부터 정렬함

flex-end: 자식요소를 주축의 종료지점에 배치함.

※ 세부 컨텐츠의 순서 및 정렬에는 관여하지 않고, 한 덩어리의 컨텐츠를 어디에 놓을지만 결정함

* flex: 1;

= flex-grow: 1; / flex-shrink: 1; / flex-basis: 0%;

flex-basis의 속성이 너비, 높이값을 무시하는 것이므로 flex에 숫자값을 부여하면 width값을 무시하고 전체 페이지의 가로 너비에서 설정값만큼 나눠가짐

css 선택자 종류와 예시

전체 선택자: ' * '

아이디 선택자: ' # '

클래스 선택자: ' . ' (태그의 종류나 위치가 달라도 같은 클래스가 부여됐을 시 동시에 호출)

이중 조건 선택자: ' .txt.text ' (두가지 클래스를 동시에 만족하는 요소를 호출)

자손 선택자: ' .div .child_div '

태그 선택자(요소 선택자): ' div ' (해당 페이지 안에 있는 모든 div요소를 호출, 사용에 유의해야함)

직계 자식 선택자: ' > ' ' .txt > p ' (단순한 태그 선택자로는 한계가 있을 시 직계 자식 선택자 ' > ' 사용으로 직계 자식 호출이 가능)

복수 선택자: ' , ' ' .p, .s ' (and의 의미)

상태 선택자: ' : ' ' input:checked ' (input 태그가 checked 상태일 때를 의미함)

인접 형제 선택자: ' + ' (어떤 요소가 특정 상황이나 조건일 때 바로 뒤에 있는 형제 요소를 호출하는 것으로 바로 다음에 오는 요소만 적용 가능함. 중간에 다른 요소가 끼어있으면 안됨)

일반 형제 선택자: ' ~ ' ' h3 ~ ul ' (상속받은 요소에 적용, 예시에서 h3에는 적용되지 않음)

가상 선택자: ' :before ',' ' :after '

속성 선택자: ' [ ] ' ' input[type='email'] ' (' a[target] ' < a태그 중 target 속성이 존재하는 요소만을 선택)

* 속성 선택자 종류

h2[class^='main'] : main으로 시작하는 클래스를 가진 요소를 호출

h2[class$='_txt2'] : _txt2로 끝나는 클래스를 가진 요소를 호출

h2[class~='com'] : com이라는 클래스 명을 앞뒤 여백(띄어쓰기)을 가지고 완벽히 포함하는 요소를 호출

h2[class*='txt'] : txt라는 클래스 명을 포함하는 모든 요소를 호출 (띄어쓰기 상관없이)

css 선택자의 우선순위 : !important > html style > id > class > 태그 > 상속된 속성

기본적인 규칙 : 나중에 쓴 내용이 이전의 내용을 덮어씀

1. 속성의 값 뒤에 **!important** 를 붙이면 무조건 최우선이 되지만, 절대 사용해서는 안됨 (나머지의 우선순위가 다 꼬이게 되어 적용이 어려움. 따라서 정말 최종적으로 사용할 수 밖에 없을 때만 사용)

2. **html에 style 속성**으로 css를 적용하면 (최)우선이 됨, 그러나 html 문서의 길이가 너무 길어지고, html과 css에 스타일을 나눠서 적용하면 협업과 유지보수가 불편해지므로 되도록 지양함

3. **id**로 지정한 속성 4. **class**로 지정한 속성 5. 상위 객체로부터 **상속된 속성**이 적용

클래스 표기법

스네이크 표기법: menu_web

카멜 표기법: menuWeb

파스칼 표기법: MenuWeb

헝가리안 표기법: StrMenuWeb (앞에 변수의 목적이나, 자료 형태를 적어주는 방법)

표기법이 중요한 이유: 아이디와 클래스 사용 빈도가 매우 높기 때문

개발자들 간의 룰

1. 클래스 남발하지 않기

=> 되도록이면 태그 그대로 사용하고 css호출은 태그나 속성 선택자 등으로 호출하여 클래스 공해를 줄이는 것

2. 규칙이 있는 클래스 사용하기

=> bem 방법론(block elements method) 등 클래스 명을 짓는 규칙을 사용하여 되도록이면 의미있고

유지보수에도 도움이 되도록 해야 함 ex) header_nav_menu-web

대상을 안보이게 하는 방법

1. display: none;

: 공간을 차지하지 않고 기능도 사라짐

2. opacity: 0;

: 공간을 차지하고 상호작용도 가능하나 보이지만 않음

3. visibility: hidden;

: opacity와 흡사함, 공간은 차지하지만 상호작용은 안됨

4. appearance: none;

: display: none; 과 opacity: 0; 의 중간 형태로 존재는 하지만 공간은 차지하지 않고 기능은 사용 가능

data- 속성의 개념

보통의 속성 값들은 DOM에 영향을 주나, data- 속성은 DOM에 영향을 주지 않음

따라서 개발자는 data- 속성을 이용하여 특정 값을 저장해두면 DOM에 구애받지 않고 사용 가능

transform 관련 css 속성

※ transform은 한 번에 여러 효과를 적용시킬 수 있으나, 중첩 시 작성한 순서에 따라서 결과가 달라짐.

그러므로 원하는 효과를 얻기 위해서는 적용시키는 순서에 주의해야함

* translate: 이동 효과, 방향에 양수 혹은 음수가 적용됨

* scale: 축소 또는 확대하는 것으로 1보다 큰 값이면 확대, 1보다 작은 값이면 축소되며 값은 하나만 적음

scale은 모든 방향으로 적용, scaleX는 x축으로만 적용, scaleY는 y축으로만 적용

* rotate: 회전 효과이며 단위는 deg로 사용

* skew: 왜곡 효과, 해당 도형이나 개체를 비틀어서 기울이는 형태로 만듦. skew값을 90deg를 줄 경우 개체가 아예 사라진 것처럼 보임 (skew 자체를 잘 안쓰지만 특정 상황에서 더러 사용함)

* perspective: 원근 효과의 값을 조절하는 것으로 700~1600px 정도의 값을 추천함

* transform-origin: 기준점을 잡는 것으로 left center 등의 방향으로 설정 가능

transition 관련 css 속성

※ transition은 모션이나 애니메이션 속도를 조절하는 것으로, 부여한 속성이 '즉시' 결과값으로 나타나는 것이 아닌, '일정 시간에 걸쳐 자연스럽게' 나타나도록 하는 것을 뜻함

* transition-duration: 지속시간을 의미하며 단위는 s를 사용(ms도 가능하지만 사용하지 않음)

* transition-property: 적용시킬 대상을 의미하며 보통 all이라고 적고, 필요에 따라 특정 대상을 적음

* transition-timing-function: 가속도를 의미함

linear: 처음부터 끝까지 일정한 가속도 (자주 사용)

ease: 천천히 시작되어서 빨라지다가 마지막에 다시 느려짐

ease-in: 천천히 시작해서 노멀하게 마침

ease-out: 천천히 마침

ease-in-out: 천천히 시작해서 천천히 마치는 것

* transition-delay: 지연시간을 의미함

animation 관련 css 속성

※ animation: 사용자가 어떤 동작을 하지 않더라도 미리 지정한 조건에 맞게 자동으로 반복하는 효과

@keyframes를 사용하여 애니메이션 세트를 지정

animation-name: 키프레임으로 등록한 모션의 이름을 호출

animation-duration: 지속시간, 키프레임 모션의 한 세트를 얼마동안 동작할 것인지 지정

animation-timing-function: 가속도 설정으로, 보통은 linear로 부여하고 추가적인 효과는 cubic-bezier에서 조정

animation-iteration-count: 재생 횟수, 보통 infinite로 설정

animation-delay: 지연시간, 없는 것이 기본값

* animation-direction의 값

normal: 시작지점에서 종료지점이 하나의 세트로 지정

alternate: 시작~종료~시작으로 하나의 세트가 지정

alternate-reverse: 종료~시작~종료로 하나의 세트가 지정

※ hover 시에 animation-play-state: paused 혹은 running을 사용하여 모션의 동작 상태 제어 가능

* animation: name duration timing-function (delay) iteration-count (direction);

=> 한 줄로 표기 가능

<form> 태그의 하위 태그

<fieldset>: form의 구역을 나누는 '필수' 태그

<legend>: 필드셋 태그 안에 반드시 작성해야하는 태그로 해당 필드셋에 어떤 내용을 담고 있는지에 대한 제목 태그

※ <textarea> 태그 사용 시 자동으로 생성되는 cols와 rows는 css에서 작성할 것이기 때문에 지우고 사용하면 됨

position: fixed 와 position: sticky

position: fixed의 기준점은 '뷰포트' (현재 보고있는 전체화면)이며

스크롤에 영향을 받지 않고 지정된 위치에 계속 있음 (높이값 인식 X)

position: sticky의 기준점은 '직계 부모'이며

스크롤이 필요 없을 때 처음의 자리를 고수하고, 스크롤이 내려가거나 화면에서 사라질 땐 지정한 곳에 위치함.

그리고 처음의 화면으로 돌아가면 처음의 자리로 돌아감

흔히 sticky는 '코드의 흐름에 따라 배치한다'고 표현하며

특정 상황(대개 스크롤)이 되면 해당 대상을 브라우저에 고정시킬 수 있음

웹 페이지 or 웹 어플리케이션의 형태 구분

1. 반응형 웹사이트

: 사용자의 디바이스에 따라 미디어쿼리가 적용된 css가 보여짐으로써 같은 코드로 다양한 디바이스 환경에 같은 느낌의 홈페이지를 경험할 수 있게 하는 구현방식

장점: 거의 동일한 html의 코드로 구현하기 때문에 2개의 디바이스를 커버할 경우 약 1.5배의 노동력이 듦(비용 절감)

단점: 하나의 디바이스에 전문적인 UI나 컨텐츠 소화 능력은 부족함, 따라서 UX의 만족도를 최상으로 충족시키기엔 한계가 있음

2. 적응형 웹사이트 (주소가 2개 이상)

: 미디어쿼리를 사용하지 않고, 디바이스에 따른 모바일, 태블릿, 데스크탑에 완전히 다른 주소와 사이트를 만들어서 최상의 UX를 제공하는 구현방식

장점: UX를 최상으로 끌어올릴 수 있음

단점: 비쌈

미디어쿼리의 개념

기본적으로 웹 버전에 작성한 코드는 유지되나, 미디어 쿼리 부분에 코드를 작성하면 기존에 있던 값에 덮어쓰게 됨 = 미디어 쿼리에 작성하지 않은 부분은 웹 버전의 코드 내용으로 유지됨

미디어 쿼리 부분에만 추가한 것은 해당 영역에서만 적용되며, 만약 미디어쿼리 구간을 두 개 이상으로 설정했을 시 상위 버전에서 적용한 것은 하위 버전에도 유지되므로 추가된 내용을 삭제하고 싶다면 초기화 작업이 필요함

초기화하는 대략적인 방법

margin-left: 30px > margin-left: 0px로 덮어쓰기

position: absolute > position: static으로 초기화

미디어 쿼리 웹 버전에서 top 좌표를 부여 했는데 모바일에서 bottom 좌표를 부여해야 한다면

top: auto를 주고 bottom 값을 부여하기

반응형 웹사이트, 미디어쿼리 적용 시 규칙

1. 모든 디바이스 전용 UI에 대한 내용은 웹 버전 html에 코딩되어야 함

-> 이후 디바이스 별 세부사항은 따로 적용

예) 모바일 전용 햄버거 버튼의 경우에도 웹에서 코딩하고 보이지 않게 처리한 뒤 미디어쿼리의 해당 디바이스 영역에서 보이게 후처리를 함

2. 웹 버전에서 고정 픽셀을 적용했던 부분을 미디어쿼리에서는 %나 뷰포트로 변환해야함

예) 웹 버전에서 width: 1000px; 이었다면 미디어쿼리 구간에서는 100%로 변경하는 식

뷰포트의 개념 : vw, vh, vmax, vmin

vh: 브라우저의 세로폭을 백등분한 단위로 디바이스의 높이가 1000px이라면 1vh는 10px인 것

vw: 브라우저의 가로폭을 백등분한 단위

vmax: 브라우저의 긴 폭을 기준으로 백등분한 단위

vmin: 브라우저의 짧은 폭을 기준으로 백등분한 단위

SCSS에서의 @for 반복문

@for 변수 from (시작 숫자) through (끝 숫자){...} 의 형태로 작성하며,

through의 자리에 to가 올 수도 있음

through는 뒤에 오는 숫자 '이하'로 반복되는 값을 사용하며

to는 뒤에 오는 숫자 '미만'의 반복되는 값을 사용함

SCSS의 mixin

scss에서 자주 사용하는 코드를 재사용할 수 있도록 하는 방법

기본값으로 코드를 만들고 믹스인 호출 시에 대체값을 넣어서 사용하는 것도 가능함

ex) _mixin.scss 파일에

@mixin button($fontSize: 12px, $wid: 80px) {

font-size: $fontSize; width: $wid; } 를 입력하고

style.scss 파일에 사용하고 싶은 구간에다가

@include button($wid: 150px);

이런식으로 입력하면 ( ) 안에 쓰이지 않은 폰트사이즈는 믹스인에서 설정한 대로 되고

( ) 안에 입력한 값만 대체되어 나타남

웹 표준, 웹 접근성 권장사항

섹션 태그에는 해당 섹션의 주제를 클래스나 아이디로 표현해야 함

클릭이 가능한 대상은 클릭을 할 수 있다는 정보를 전달하기 위해 cursor: pointer; 를 사용해주어야 함 (이것과 반대 비슷한 개념은 pointer-events: none이 있으며 opacity나 visibility도 사용)

<table> 태그 사용 시 <caption> 태그를 사용하여 해당 테이블 태그가 어떤 내용인지를 필수적으로 작성해야함 (제목 형식)

<select> 태그 사용 시 속성값 채우지 않으면 유효성 검사에서 에러가 뜨므로 꼭 채워야 함

SEO (검색 엔진 최적화)

: 사이트의 컨텐츠 페이지 정보가 검색에 상위에 노출될 수 있도록 최적화하는 과정

컨텐츠의 본질 중요성을 판단하기도 하고, 같은 조건의 컨텐츠일 경우 메타태그의 작성이 우선순위 영향을 줌

* index (색인)

어떤 포털 사이트든지 모든 페이지를 색인이라는 곳에 저장함

즉 어떤 페이지를 만들어서 서버에 올리면 그 페이지에 포털 사이트가 고유의 url 을 입히고 url 색인이라는 장소에 저장함

* crawling 크롤링

크롤러(포털 사이트에서 색인을 담당하는 봇)가 해당 페이지에서 정보를 추출하는 행위를 뜻함

만약 새로운 페이지 혹은 업데이트 된 페이지가 발생하면 크롤링을 거쳐서 해당 페이지가 index 에 저장되는 것

* web crawler 크롤러

크롤링을 하는 자동 소프트웨어(봇)

* 오픈 그래프 (og)

어떤 html 문서의 메타정보를 쉽게 표시하기 위해서 메타정보에 해당하는 제목, 설명, 문서타입 등을 사람들이 통일해서 쓸 수 있도록 정의해놓은 약속(프로토콜)

페이스북(메타)에 의해서 기존의 다양한 메타 데이터 표기 방법을 통합하여 만들어졌고 처음에는 페이스북에서만 사용하다가 이후 모든 곳에서 사용하게 됨

보통 어떤 내용을 눈에 게시할 때 나타나는 내용임 (카톡으로 전송 시에 뜨는 그것)

=> 웹 표준을 위해서는 시멘틱 태그로 마크업 하는 것이 중요함 !!