홈페이지의 inspect, 요소 검사 -> f12 통해 볼 수 있음

self closing 태그들 마지막 공백+역슬래쉬는 HTML버전에 따라서 필용할수도 없을수도 있지만 무조건 해주는게 좋다 !

label 태그는 input 태그와 같이 써야 함.

label 태그의 for attribute 와 input 태그의 id attribute에 같은 값을 넣어야 함

그럼 label 클릭이 for과 같은 id를 가지고 있는 input을 실행시킴

id는 어떤 태그든 가질 수 있는 attribute

id는 unique identifier임. element당(태그) unique한 하나의 id만을 가져야 함. (그래야 정상 작동함) 앞서 말했듯이 브라우저는 에러같은거 안냄. 걍 원하는대로 작동을 안할뿐

그리고 또 중요한 이유가 css가 id를 통해 contents의 how를 설정하기 때문임.

가장 많이 보게 될 태그 중 하나가 div 태그.

div은 tag들을 구분시켜주는 것이라 생각하면 됨. div박스안에 원하는 태그들을 넣는 것.

기본적으로 줄바꿈(옆에 다른게 오지 않음) 그리고 semantic하지 않음 (보는것만으로 무슨뜻인지 알 수 없음). 애초에 의미를 별로 가지고 있지 않음. 그냥 박스안에 담는거니까. 이게 무슨 박스인지 div태그만을 보고 알기는 힘들겠지? 예전에는 semantic한 녀석들이 없어서 웹사이트들이 div로 뒤범벅이였음.

semantic한 태그는 코드 자체로 의미가 부여되어 있는 녀석들임.

header태그. head와 헷갈리면 안됨. header는 body안에 들어가는 태그임. 기본적으로 div태그를 대체할 수 있음. 그리고 얘는 semantic임. 왜냐면 header는 말그대로 header컨텐츠를 담고있을거라는걸 보자마자 알 수 있으니까

즉 div id=”header” 라고 써도 되지만, header라고 쓰는게 의미를 전달하기에 편하다는 것 같음.

main태그. 마찬가지로 div id=”main” 해도 되지만 main이라고 쓰자. 이 파트가 해당 사이트의 main에 해당하는걸 semantic하게 전달해 주니까

footer태그. 꼬릿말을 위한 태그. 마찬가지로 div id=”footer”이나 동일 (애초에 div 라고 해도 브라우저 상에서 보이는 것은 동일함. id를 사용해서 뭔가를 하는게아니라면)

p태그. paragraph. 단락이나 절 수준의 text를 사용할 때. 물론 이안에 짧은 글을 넣어도 오류는 생기지 않음. 다만 semantic한 의미를 생각해보면 적절하지 못함

span태그. 한문장 수준의 짧은 text를 사용할 때. 마찬가지로 이 안에 긴 text를 넣어도 문제는 없음. 태그가 가지고 있는 semantic한 의미와 다른 형태로 써버렸을 뿐임.

이외에도 많은 semantic한 태그들이 존재.

즉 코드를 짤 때부터 semantic하게 짜면, 올바른 태그를 사용하면 보고 이해하고 파악하는게 빨라짐.

당연하게도 모든 태그를 외우는 것은 불가능. 많이 쓰는 태그들은 저절로 외워지는 거고 모르는 태그는 그때그때 검색해서 쓰면됨. 이런 태그가 있지 않았었나? 하는 정도로만 알고 있어도 괜찮다는 뜻이지.

CSS

우선 CSS가 어떻게 HTML과 섞이는가? 크게 2가지 방법이 존재.

첫번째는 원시적인 방법으로, HTML파일 안에 HTML코드와 CSS코드를 다 넣어두는 것

두번째는 좀더 추천되는, professional한 방법으로 HTML과 CSS를 분리

첫번째 방법으로 css를 사용할 때 (inline css)

style태그를 사용함. style태그는 head태그안에 있어야 함. 그리고 CSS코드를 style태그 안에 작성하면 됨.

두번째 방법으로 css를 사용할 때, css파일을 include, (external css)

css파일을 만듬. 그리고 html파일에서 link태그를 이용하여 css파일과 연결

이 방법이 좋은 이유는 css파일을 다른 html파일에 연결시켜 쓸 수 있음. css가 길어질 때 html문서 보기쉬워지기도 하고

css파일에서 코드 작성할 때 크게 3가지를 염두에 두어야 함

1번은 css가 하는 일은 HTML태그를 가리키는 일이라는 것. 이때 가리키는 것 자체를 selector 라고 함. 그리고 그 태그가 크기가 뭐고 색이 뭐고 등등을 property 라고 함. selector가 많은 property를 가질 수 있고, 중괄호로 묶어서 씀.

즉 css는 HTML의 태그를 잡아와서 원하는 형태로 바꾸는 것.

태그처럼 속성도 매우 많고 다 외울 수 없음. 찾아서 써야함 그때그때.

다만 css속성은 지켜야 하는 문법이 있음.

속성: 값;

형태로 사용

참고로 밑줄 \_ 이나 슬래쉬 / 공백 같은거 속성이름에 안쓰임 – 가 쓰임

css에서 selector할 때, 태그만으로 select하면 해당 태그를 전부 지칭하는 것이 됨. 해당 태그가 여러 개가 존재한다면 특정 태그만을 가리키기 위해 id를 사용하면 됨.

#id { } 해서 중괄호 안에 css를 작성하면 됨.

selector 여러 개에 대해서 한번 css 작성 가능

selector1, selector2, …. {

} 이런식으로 작성하면 해당 selector가 전부 저 css를 받음

css에서 box는 매우 중요하고 많이 쓰임. 참고로 박스는 기본적으로 옆에 아무것도 올 수 없음.

옆에 다른 element가 못 오는걸 block이라 부름 (box보다 block이 정확한 명칭)

body, div, p, 등이 해당함. 대부분의 box는 블락이라고 생각하면 됨.

박스가 아닌것들은 하나의 element 옆에 다른 element가 올 수 있음. inline이라 부름 (in the same line의 준말). block이 아닌 것들, inline인 것들이 소수임.

span, a, image 등

하지만 당연하게도 inline과 block은 바꿀수 있음. inline이 bolck이 되게하는거나 그 반대나 다 가능. 이걸 담당하는 property가 dispaly

inline은 높이와 너비가 없음. block은 높이와 너비가 존재.

html태그에 css넣으면 웹사이트 전체가 영향을 받음.

브라우저가 element에 자동으로 부여하는 많은 속성들이 존재.

박스가 가지는 3가지 큰 특징 (속성)

margin 과 padding과 border

border은 box의 경계

margin은 box의 border의 바깥에 있는 공간

margin 적용할때 값 하나만 전달 -> 상하좌우 전부 해당 값. 값 두개 전달 -> 첫번째 값이 상하, 두번째값이 좌우. 값 네개 전달 -> 위, 오른쪽, 아래, 왼쪽, 시계방향순으로 설정됨.

collapsing margin : margin의 상하에서 일어나는 현상. 포함되어 있는 박스가 포함하는 박스의 경계와 같은 상황에서 (이 상황이 되면 두 박스의 마진이 하나가 되는 것과 같아짐 ) 포함되어 있는 박스가 포함하는 박스의 마진을 넘어설 때.

padding 은 box의 border의 안쪽에 있는 공간

값 전달 방식 -> margin과 동일.

border는 말그대로 box의 경계. border에 여러 종류 있긴 한대 대부분 잘 안씀.

border style mdn 검색하면 border의 속성 볼 수 있음. border에 값 전달할 때 순서가 있음, 크기, 스타일, 색상 등등 순서에 맞게 값 전달해야 함.

태그로 호출하거나 / id로 호출하거나 / \* 로 호출하거나 -> \* 로 호출하면 모든 녀석들을 호출함 \*은 전체를 의미

border은 inline에도 적용이 된다.

inline은 높이와 너비를 가지지 않음. 그래서 위, 아래에 margin을 가질 수 없음. 다만 padding은 위아래 좌우 다 가질수 있음. 따라서 위 아래 마진을 가지고 싶으면 해당 inline element를 block으로 바꾸어 주어야 함.

class -> id가 유일해야 한다는 규칙에서 발생할 수 있는 단점을 해결하기 위한 방법 중 하나.

10개의 p 태그 중 5개의 p 태그에 적용하고 싶은 css가 있다면 ? p로 적용하면 10개가 다 적용될거고, id로 적용할려면 하나하나 일일이 적용해야 함. 5개의 p에다가 같은 클래스를 적용하고 해당 클래스의 css를 작성하는 것.

즉 id가 element를 유일하게 지칭하는 방법이라면, class는 element를 겹쳐도 되게 지칭하는 방법.

class를 지칭하는 방법은 .class명

.class1{

}

이런식으로 쓰면 class1에 해당 css적용

class는 여러 개 가질 수 있음. 여러 개 클래스 적용할 때 그냥 스페이스바로 한칸씩 띄워서 적음 되네

inline이 높이랑 너비를 가지게 하고 싶다면 -> inline-block 으로 만들면됨. inline-block은 높이, 너비, margin 가지고 있으면서 다른 요소가 옆에 올 수 있음.

다만 inline-block은 responsive design을 지원하지 않음. (따라서 창크기나 모니터 크기등에 영향을 받아서 보이는 화면이 달라질 수 있음)

flexbox. 사용시 지켜야 할 3개의 규칙이 있음.

첫번째 : child element에는 어떤 것도 적지 말아야 한다. parent element에만 적어야 함. 이때 부모는 flex container가 되어야 함. 즉 부모의 display속성이 flex가 되어야 한다.

이후 부모한테 여러가지 속성 넣어서 child를 바꿀 수 있음.

justify-content 속성을 통해 화면에 나타나는 위치 조정가능. 중앙정렬이나 우측정렬이나 간격 나란히 놓기 (space-evenly)등등.

부모와 자식관계 : 태그1을 열고 태그1안에 다른 태그2가 들어올 때 태그1이 부모, 태그2가 자식.

예를들어 <body> <div> </div> </body> 형태면, body가 부모 div가 자식.

두번째 세번째 -> flex container의 축과 관련되어 있음. main axis(주축) 와 cross axis(교차축) 두가지 축이 존재. 기본은 주축이 수평, 교차축이 수직인데 바뀔 수 있음 ( flex-direction 속성은 default가 row). flex 속성들은 이 두가지 축 중 특정 하나의 축에 적용됨 보통. 그래서 해당 축에 맞는 속성에 값을 전달해서 원하는대로 디자인 하면 됨.

justify-content는 주축에 적용 / align-items는 교차축에 적용

값은 px 값만 있는게 아니라 다양한 값이 있음. vh는 view height (%)를 의미해서, 화면에서 몇퍼센트를 차지할 것인가를 나타낼 수 있음.

flex는 보통 높이나 너비를 초기 사이즈로만 생각하고 화면에 따라 보이는 사이즈를 바꾸는데, 사이즈를 유지하게 하는 방법 있음. flew-wrap 속성 같은거 참고.

position 속성

fixed 값 -> 화면에 항상 고정된 위치에 보임. 스크롤하더라도. 모든 레이어 최상단에 위치 (홈페이지 위쪽에 존재한다는게 아니라, 레이어가 높다는 것은, 하위 레이어의 컨텐츠를 가리고, 모든 녀석들을 덮고 노출될 수 있다는 뜻)

default값은 static임. 레이아웃이 박스를 처음위치에 두는 것.

relative값을 전달하면, top, left, 등의 속성을 사용할수 있는데, element가 처음 위치한 곳(static일때)을 기준으로 상하좌우로 움직일 수 있음.

absolute값을 전달하면 차근차근 부모 중에 relative가 있는지 찾아서 그 부모 기준으로 움직임. relative한 부모 못 찾으면 viewpoint기준.

pseudo selector

element select를 조금 더 디테일하게 해주는 방식

지칭할 때 뒤에 : 붙이면 여러가지 옵션이 나옴. 그 옵션에 해당하는 녀석을 지칭 가능 ( 좀 더 디테일하게 특정 녀석이 지칭이 되겠지? )

first-child, last-child같은 옵션뿐만 아니라 nth-child(n) 같은 옵션도 존재. 심지어 n에 숫자만 넣는게 아니라 even, odd같은 녀석 넣을수도 있음. 게다가 2n, 2n+1, 3n 이런식으로 점화식을 넣어도 잘 작동된다. 사실 even odd 보다도 점화식 넣어서 하는걸 익히는게 더 좋은듯

이런방식으로 css를 작성하면 html을 건드려서 뭐 특정 녀석에 id를 넣거나 클래스를 넣거나 하는 짓을 안해도 css만으로도 원하는 녀석을 지칭해서 styling가능.

combinator

마찬가지로 element select를 조금 더 디테일하게 해주는 방식의 하나.

어떤 태그(부모) 내의 특정 태그(자식)에 대한 select 방법

부모태그 자식태그 { } 이런식으로 지칭하면 먼저 부모태그를 찾고, 그 부모태그 내에 있는 자식태그를 찾아서 그 자식태그의 css작성가능. 이런식으로 지칭하면 부모 태그의 바로 자식태그가 아니라도 자식의 자식이든 안으로 쭉쭉 찾아서 다 지칭함

부모 바로 밑에 있는 자식까지만 찾겠다면

부모태그 > 자식태그 {} 이런식으로 지칭하면 됨. 이러면 자식의자식까지 안찾고 딱 direct한 자식태그중에서만 지칭.

같은 부모 내에서 있는 형제들끼리 지칭할때는

형제태그1 + 형제태그2 {} 이런식으로 지칭하면, 형제태그1 바로 다음으로 오는 형제태그 2를 지칭할 수 있음. (바로 안오고 중간에 다른 형제가 껴있으면 지칭 안됨)

형제태그1 ~ 형제태그2 {} 이런식으로 지칭하면 형제태그2가 형제태그1 다음 바로 안오더라도 형제태그 2를 찾아서 지칭함. 그리고 형제 태그2가 몇 개가 있던간에 (그 중간에 다른 태그들이 껴있어도 상관없이) 형제태그 1 이후로 오는 모든 형제태그 2를 지칭함.

이외에도 수많은 pseudo selector가 존재.

attribute를 가지고 select 가능

태그1: 속성1 {} 이런식으로 지칭하면 태그1 중 속성1을 가지는 녀석들을 모두 지칭할 수 있음.

태그2[속성2=”속성값2”] {} 이런식으로 지칭하면 태그2 중 속성2의 속성값이 속성값2인 녀석들을 모두 지칭할 수 있음.

태그3[속성3 ~= “name”] {} 이런식으로 지칭하면 태그3중 속성3의 값 중 name값이 포함된 element를 지칭. (주로 문자열을 속성값으로 가지는 속성에 쓰겠지?)

태그3[속성3 $= “name”] {} 는 name으로 끝나는 녀석을 지칭

태그3[속성3 ^= “name”] {} 는 name으로 시작하는 녀석을 지칭

states

여러가지 state들이 존재. 예를들어 active같은 경우는 커서가 해당 element를 클릭 중인 상태를 의미.

태그: active{} 이런식으로 active일때의 css를 지정할 수 있음.

hover는 커서가 해당 element위에 있을 때 적용.

focus는 키보드로 선택되었을 때 적용.

focus-within은 해당 태그의 자식 중 어떤것이라도 focused되었을 때 적용.

visited는 링크에만 적용. 링크를 방문한적이 있는 상태면 적용

그리고 이런 지칭은 다른 지칭과 조합해서 사용 가능

태그1: hover 태그2{} 이렇게 지칭하면 태그1이 hover상태일 때 자식 중 태그2를 지칭.

태그1:hover() + 태그2{} 이렇게 지칭하면 태그1이 hover상태일 때 바로다음으로 오는 형제 태그2를 지칭.

pseudo element

element는 아니지만 css를 통해서 스타일링 할 수 있는 애들이 몇 개 있음

예를들어 input의 placeholder같은 경우

input::placeholder {} 이런식으로 placeholder를 지칭해서 스타일링 가능.

p::selection {} 이런식으로 하면 해당 p문장에서 드래그한 녀석들, 문자들을 스타일링 가능

p::first-letter {} 하면 첫번째 문자의 스타일링 가능.

예시에서는 input, p 같은거만 썻는데 다양하게 있음. span에도 당연히 first-letter나 selection 있겠지?

css에서 color 값 표현 방법

이름을 가진 컬러는 이름으로 가능.

16진수 표현방법 #으로 시작해서 6글자.

RGB표현방법. (0~255 까지 숫자 3개)

rgba표현방법. rgb + a(투명도. 0~1사이값) 0이면 완전투명해서 안보임.

root element에 변수 추가하기. custom property라고 하는데 variable과 같은맥락

:root는 기본적으로 모든 document의 뿌리가 됨.

<style>

:root{

변수명 : 값;

}

</style>

이런식으로 root안에 변수명과 변수값을 저장 가능.

해당 변수를 쓰고싶을때는 값을 넣는 자리에

var(변수명) 하면 해당 변수값으로 사용가능.

참고로 변수명은 --원하는변수명 으로 써야되고 중간 공백 당연히 안되고 근데 --포함해서 전부 다가 변수명임. -없는걸로 착각하면 안됨. 무조건 dash 2개 가지고 있어야됨 변수명은

그리고 변수가 가질수 있는 값은 한 개만 있다고 생각하면 안됨

border 속성값에 쓸 변수라면

--border-var : 1px solid red; 이런식으로 변수값 가져도 된다는 것.

그리고 변수값에 변수써도 됨

--main-color : red;

--border-var : 1px solid var(--main-color); 이런식으로도 얼마든지 쓸 수 있다.

고급 css

transition 이란 어떤 상태에서 다른 상태로 가는 것을 애니메이션화 한 것.

transition 속성은 state가 없는 요소에 붙어야 함. 정확히는 그 element가 처음 스타일을 받는 곳,기본상태인 곳, transition 되기 전의 스타일을 나타내는 곳에 있어야함

만약 state 붙은 요소에 넣으면 기본상태에서 -> state로 변할때는 적용되는데 state에서 기본상태로 돌아갈때가 적용이 안됨. 물론 그렇게 쓰고싶다면 붙이면되긴하지.

transition : transition시킬속성1 시간 방식 등등, transition시킬속성2 시간 방식;

이런식으로 작성 가능

또는 transition: all 시간 방식 ; 이런식으로 쓰면 변화가 있는 모든 녀석들에게 적용됨

방식에는 linear, ease-in, ease-in-out, ease-out, ease 등이 있음.

transformation : 한 요소를 변형시키는 것. 돌리거나 짜부러뜨리거나 옮기거나 이런거 가능

transform: 속성1(속성값1) 속성2(속성값2) 뭐 이런식으로 작성

transform: scaleX(3) rotateX(45deg) translateX(-100px) 등등

그중에서 translate 를 이해하는게 중요. 얘는 margin이나 padding같은거를 고려하는게 아님. 다른 요소의 box를 변형시키지않고 원하는 요소만 이동시킴. transformation은 box차원에서 일어나지 않음.

transition 과 transform을 조합해서 사용할 수 있음.

태그 {

transition : transform ~~

}

태그 : hover{

transform : ~

}

이런식으로. hover state일 때 transition 이 일어나는데 일어나는 transition이 transform

그리고 역시 transform mdn 검색하면 다양한 transform 종류 볼 수 있음

Animation

transform이나 transition같은거랑 상관없이 그냥 애니메이션 만드는 것.

@를 적으면 애니메이션으로 인식. keyframes라는 애니메이션을 쓴다면.

@keyframes 내애니메이션의이름 {

keyframe의 첫번째 옵션 {

애니메이션 내용;

보통 transform 을 많이 씀.

}

두번째 옵션{

}

그다음 옵션

… 이런식으로

}

그리고 애니메이션을 넣을려는 태그에

태그 {

animation : 내애니메이션의이름 실행시간 실행방법 실행횟수 등등 ;

}