**HTML 공부 정리**

**전채현**

**인터넷과 웹의 차이**

인터넷은 인터 네트워크 기술 중에 하나이다

-서로 다른 제조사의 컴퓨터끼리 자원을 주고 받기 힘들었다 ( 각 네트워크 안에서만 가능했다)

-네트워크와 네트워크를 연결하기 위해서 inter network가 나온 것이다

-> 미 국방부 산하 Arphabet에서 internet이라고 부르기로 함

이제 서로 자원을 주고 받을 수 있다

클라이언트 : 요청하는 쪽

서버 : 제공하는 쪽

프로그램들이 요청하는 방식에 따라서 서버와 통신한다 -> 표준화된 방식으로 통신해야 서로 호환이 된다

(FTP : file transfer protocol)

인터넷이 생기고 서로 주고 받기 위해서 많은 규약이 생겨난다( FTP, SMTP, TELNET, WWW(HTTP) )

인터넷이 고속도로고 WWW가 택배인것이다

===========================================

**web과 HTTP**

클라이언트에서 요청 (get 요청, Request) 웹서버에 그럼 웹서버에서 요청한 자료를 찾아서 클라이언트에게 response해준다

===========================================

**HTTP( Hyper Text Transfer Protocol )**

-> (하이퍼) 텍스트를 전송하는(주고받는) 프로토콜 (규약, 규칙이나 약속)

- 정보를 텍스트 형태로 저장하여 보관했음

- 네트워크의 발달로 인해, 텍스트 형태의 정보(information)를 주고받기 위한 프로토콜

- 텍스트 + 미디어(비디오, 오디오) 를 전달 가능한 형태로 변경

- 어떻게 이미지나 오디오나 비디오를 표현할 것이가 에 대한 규칙이 필요해짐

- 태그라는 형식으로 텍스트, 이미지, 오디오, 비디오를 표현

- HTML( HyperText Markup Language) -> 다른 프로그램들이 읽어서 객체화 할 수 있다

- HTTP를 통해서 전달하는 웹 문서는 HTML 규칙에 맞게 작성되어 있다

- HTML은 태그라는 형태로 구성되며, 화면에 표현하고 싶은 요소를 태그 종류별로 나타낼 수 있다

- 초창기의 HTML 문서는 항상 고정된 형태를 나타내는 문서이다( Static Web )

- 세션, ip, 쿠키, 캐시 등에 의해 서로 다른 내용을 나타내는 웹 형태 ( Dynamic Web )

- Dynamic Web을 구성하기 위해서 프로그래밍 언어가 필요하다(C, Perl, Java, Python)

-> 왜 프로그래밍 언어가 개입하게 되었는지 ??

--> 텍스트 기반이며 글자를 사용하는 규칙을 익혀야 한다

=> 네트워크

- Internet : 서로 다른 (소규모) 네트워크들을 통합하여 연결한 거대한 네트워크

- 인터넷 상에서 용도에 따라 서로 다른 프로토콜(규약)로 통신을 주고 받을 수 있다

- 그 중 웹 문서를 주고받기 위한 규약을 HTTP이다 ( TCP, IP, UDP )

- WWW ( World Wide Web )

- IP ( Internet Protocol ) : 인터넷을 사용하기 위해 지켜야 할 규칙이나 약속

- IP Address : IP규칙대로 서로 통신하기 위해 컴퓨터가 가지는 논리적인 주소

- 컴퓨터는 숫자 체계로 각 컴퓨터를 식별한다

- 사람은 숫자로 구성된 체계를 기억하기 쉽지 않고, 의미부여도 쉽지 않다

- 숫자를 대체할 수 있는 글자 형태의 주소 체계가 필요하다

- 영역에 이름을 부여하여 관리하는 시스템 ( Domain Name System, DNS )

- (IP : 8.8.8.8 Domain : google.com )

=> System Server (Windows, Linux, Unix)

- Web Server 와 Web Client 로 구성된다

- 웹 서버는 문서를 가지고 있다가 클라이언트의 요청이 들어오면 문서를 보내준다

- 웹 클라이언트는 IP 혹은 DNS이름으로 웹 서버를 지정하여, 서버가 가지는 문서를 요청하고 받는다

- 웹 클라이언트는 HTML 형식으로 구성된 문서를 해석하여 화면에 표시할 수 있다 ( Web Browser )

- HTML로 문서를 만들어서 웹 서버에 올려두면, 브라우저(크롬, 엣지)로 요청하여 볼 수 있다

- 웹 서버에 올려둘 문서를 만드는 것이다 ( 도서관 시스템 )

===========================================

**웹의 3대 요소**

웹을 구성하느 3대요소로 HTML, HTTP, URL이 있다. HTTP는 웹 서버와 웹 클라이언트 간에 통신을 할 수 있도록 지원을 해주며,

URL은 웹 클라이언트가 웹 서버로 자원을 요청할 수 있게 해준다.

마지막으로 HTML은 웹 클라이언트가 웹 서버로 URL을 통한 자원 요청 후 HTML가 담긴 응답을 받으며, 웹 클라이언트 프로그램인

웹 브라우저가 이를 해석하여 사용자에게 구조화된 인터페이스를 출력시킨다.

===========================================

**Markup Language**

문서편집과 컨텐츠 구분자의 필요성

-> 아래 문자를 HyperText로 만들고 싶다

-> 무궁화를 누르면 HyperText가 되어서 관련 자료가 나오게 하고 싶다

-> 이때 Markup을 사용하여 관련 자료를 나오게 할 수 있다

-> 이러면 무궁화는 컨텐츠로 인식되지 않는다 <b>태그가 마크업하면서 의미까지 부여할 수 있다

우리나라 만세 <b>무궁화</b>삼천리 화려강산

태그의 기호(<>)가 마크업 기호로 적합한 이유

[무궁화] ,무궁화, {무궁화} -> 이런식으로 하면 다른 의미가 발생되어 다음에 괄호를 사용할 수 없다

-> 또는 중첩을 할 경우가 발생할 때 위에 기호를 사용하면 다른 곳에서 사용할 수 있는 것이 줄어든다

Markup 언어란 문서의 컨텐츠를 구분하기 위한 태그 집합

Hyper : 저 건너

안녕하세요! <b><a herf="http://www.naver.com">newlec</a></b>입니다. <font color ="blue">newlec 강의는</font>

단편적인 지식보다는 <u>용어의 올바른 이해</u>와 그것을 제대로 사용할 수 있도록 하는 <i>올바른 코딩 지칭<i>을 알려주고 있습니다.

<a> : 앵커 배가 정박하듯이 다른 문서에 닷을 내려서 연결

href : 정박용 주소

-> 하이퍼링크 : 좀 더 쉬운 문서 공의 필요성

HTML: 문서의 구조를 만들기 위한 태그 + HyperText를 위한 태그를 포함

HTML 문서의 기본 구조 : 규칙

<html> <!-- 모든 문서는 html태그 안에 존재해야 한다-->

<head> <!-- head와 body는 두번이상 나올 수 없다. 그리고 순서도 바뀌면 안된다-->

<!-- 브라우저 화면에 나오는 것이 아니다. 화면에 나올 설정을 작성한다-->

<title>first page</title> <!--문서 제목-->

</head>

<body>

<h1>Hello! HTML</h1>

<p>

<a href="www.naver.com">HTML</a> 이란

HyperText Markup Language.

</p>

</body>

</html>

===========================================

**사용하지 않을 태그에 대한 호환성을 지정하기 위한 Document Type**

HTML DTD : Document Type Definition의 종류

-> 사용 이유 : 렌더링 방식의 호환성 문제 발생

-> 다양한 브라우저의 경쟁 구도로 인한 독자적인 태그들

해결 방안 : 웹 표준 기구 W3C 탄생

===========================================

**W3C의 역할**

Draft -> Working Draft -> Candidate Recommendation -> Recommendation

(하지만 Recommendation이 강제성을 가지지는 않는다)

HTML5에서는 문서 구조의 형식을 아래와 같이 통일했다

<!DOCTYPE html>

-> 모든 브라우저에서 렌더링할 수 있도록 해달라는 뜻

===========================================

**모바일용 웹과 PC용 웹의 양분화**

-> 다양한 장치를 위한 UI구현의 문제

PC웹의 필요성 -> 작업용으로 많이 사용하고 모바일에서 소비용으로 사용

폴더 잡기

-> 문서를 구조화 하기

root

L\_\_\_\_\_\_\_ index.html

L\_\_\_\_\_\_\_ notice

L\_\_\_\_\_\_list.html

L\_\_\_\_\_\_dtail.html

L\_\_\_\_\_\_\_ member

L\_\_\_\_\_\_ login.html

L\_\_\_\_\_\_ signup-agree.html

L\_\_\_\_\_\_ signup.html

Navigation을 위한 링크 추가

/ : 절대경로

./ : 현재 자기가 있는 디렉토리

../ : 상위폴더로 가고 싶을 때

===========================================

**SGML 그리고 HTML과의 관계**

태그 사용방법에 규칙이 없다??

<meta charset="UTF-8"> -> 단일태그로 사용 닫는 태그가 없다

html 규칙에는 단일 태그는 닫는 태그를 사용하지 않아도 된다

General Markup 언어의 필요성

- 데이터를 저장하거나 전달힉 위한 마크업

-> 1960, IBM에서 범용적으로 사용할 수 있는 마크업 언어를 만듦

SGML ( Standard Generalized Markup Language )

-> 문서 타입을 정의하는 언어인 SGML의 문제점

-> 태그의 규칙(html.dtd)이 심어져 있다 이것이 기반이 되어 HTML이 만들어진다

HTML 문서의 변화

1. 시맨틱을 지원하는 문서로의 변화 - W3C

2. 프로그래밍 플랫폼으로의 변화 - WHATWG

시맨틱을 지원하느 웹의 모양

-> 데이터 자동 추출 문서로써의 변화

웹 지식 자원 <-> 온톨로지(지식표현) <-> 에이전트(소프트웨어)

웹 문서 변화의 필요성

-> 태그를 사용자가 정의할 수 있는 마크업 언어의 필요성

-> 새로운 메타 언어 : XML(규칙만 지키면 된다 / 확장 마크업 언어), XHTML

XML( eXensible Markup Language)

-> 문서 타입(형식)을 정의하는 새로운 언어의 XML

XHTML과 HTML의 버전의 역사

SGML -> XML

WHATWG( Web Hypertext Application Technology Working Group )

프로그래밍 플랫폼으로써의 변화가 요구되어 오던 웹 문서

내용에서 부터 상위로 올라가기

상위에서 하위로 내려오기 -> 경험이 쌓이면 이렇게 하고 경험이 부족할 경우 내용에서 상위로 올라가기

===========================================

**문서의 아웃라인을 잡아달라**

- 제목을 달아달라는 뜻

영역을 위한 시멘틱 태그

HTML 4 까지는 다음ㅇ과 같은 필수 구조를 설명하는 태그가 없었다

-> header, main, footer 사람들이 가장 많이 사용하고 가장 많이 추천하는 이름으로 표준화가 진행됨

-> 의미론적으로 header, main, footer를 사용

아웃라인의 계층화

-> 그룹핑되어 있는데 그 안에서 다시 나누기 위해서 등장

-> h1을 밖에서도 다시 사용하고 싶음 그래서 등장

-> 넘버링을 또 하고 싶어서

-> Section태그를 이용해 아웃라인을 잡는 새로운 방식

(section은 아웃라인에 영향을 주지 않는다)

섹션 태그들

<section/> -> <article> <aside> <nav/>

===========================================

**블록 태그들**

-> 지금까지 설명한 태그들은 모두 콘텐츠의 영역(블록)을 위한 태그들이다

블록태그는 자기 스스로가 영역을 가지고 있는 것 컨텐츠를 감싸고 있지만 자신만의 영역이 있다

기본적으로 너비는 브라우저 너비를 꽉채우는 너비 다른 컨텐츠가 옆에 올 수 없어서 밑으로 내려가는 것이다

그래서 적재되는 형태로 되는 것이다

종류에 따른 블록 스타일의 차이점

블록 스타일

제목 스타일 블록 : 블록 스타일과 비교하여 margin이 있다

목록 스타일 블록 : li 여기도 margin을 가지고 있다

-> 너비를 꽉 채우고 한층을 혼자 사용하는 것이다

-> 그래서 너비와 높이를 지정할 수 있다

인라인 태그의 기본 스타일의 이해

<div><a>열공</a><b>화이팅</b><u>응원합니다</u></div>

-> div 안에 있는 태그들은 inline 태그들이다 ( 줄을 사용하는 태그 )

<div></div>

<div></div>

블록 태그의 크기는 자신이 감싸는 콘텐츠와 상관없이 자신의 영역을 가지고 있다

인라인 태그의 크기는 자신이 감싸는 콘텐츠와 밀접하며 콘텐츠의 크기가 자신의 크기가 된다

(인라인 태그는 자기 스스로가 영역이 되지 않는다 -> 콘텐츠의 크기에 따라서 달라진다)

컨텐츠의 방을 만들어 주는 것이 블록 태그

CSS -> HTML문서를 꾸며주는 역할

style 태그를 이용한 속성의 집중화

<style>

h1{

color:blue;

}

</style>

그룹명은 class라는 선택자로 지정해서 한다

class는 다른 태그들을 그룹지어서 동일하게 속성을 지정해 줄 수 있다

아무것도 안써주면 태그로 인식하고 class로 인식하려면 앞에 . 콤마를 붙여줘야 한다

id는 엘리먼트를 식별할 때 사용되기 때문에 유일해야 한다 (unique)

id로 인식하려면 앞에 #을 붙여줘야 한다

===========================================

**콤비네이션 연산자**

Descendant selectors A B ( A태그 중에서 B태그를 찾아라 )

Child selectorA > B( A의 자식 중에서 B를 찾아라 )

Adjacent sibling selectorA + B( A에 이웃한 B를 찾아라 / A 바로 밑에 B가 있으면 적용 )

General sibling selectorA ~ B( A에 이웃한 것 중에서 B를 찾아라 / A 의 동생들 중에서 B가 있으면 선택하겠다)

===========================================

**스타일 적용 순서**

콘텐츠 블록 스타일( 가구 ) 가구를 배치할 공간이 필요하다 -> 레이아웃 블록 스타일 (가구를 배치할 공간)

집안을 대표하는 블록에 id를 붙여준다 id를 작성할 때 main-menu 대시를 사용한다 (기본적으로 )

대시와 대시를 구분하기 위해서는 언더바를 사용한다

#main-menu h1{

main-menu에서 h1을 찾으라는 의미

font-size: 20px;

}

--> 이 경우에는 main-menu안에 h1이 하나 밖에 없어서 적용이 된 것이지 h1이 하나 더 있을 경우도 발생할 수 있다

포함관계를 부모자식관계로 의인화하여 표현한 것이다 그래서 자식의 자식이 발생할 수 있는 것이다

여기서 #main-menu > h1으로 하면 main-menu 의 자식중에서 h1을 찾아라가 된다

그러니까 손자나 증손자에서 h1이 있다고 하여도 자식 중에 h1이 있는 것만 찾아서 적용하는 것이다

===========================================

**Basic Selectors**

Type selectors 엘리먼트 명

Class selectors.클래스명

ID selectors#id명

Universal selectors\*

Attribute selectors[속성=값]

#main-menu a[href="a.html"]{

font-size: 30px;

font-weight: bold;

}

--> 이런식으로 main-menu에 a태그 중 href="a.html"의 값을 갖는 태그를 찾아서 적용하는 것이다

[attr ~= value] -> 이렇게 하면 value가 포함된 것을 찾아서 적용하겠다 만약 value가 id="abcdefg"인데 abc만 적어도 적용이 가능하게끔 하는 것이다

[attr |="value"] -> value로 시작하고 뒤에 하이픈이 있는 것을 찾아서 적용하겠다

a[href^="#"] -> #으로 시작하는 것을 찾아서 적용하겠다

a[href$=".org"] -> .org로 끝나는 것을 찾아서 적용하겠다 (정규식이랑 동일함)

a[href\*="example"] -> example이 적어도 한번 이상 포함되어 있는 것을 찾아서 적용하겠다

a[href\*="insensitive" i] -> i는 ignore의 의미로 대소문자를 가리지 않는다는 의미이다 (s 도 동일한 의미)

여기서 특정 li의 값을 찾고 싶으면 id말고 class를 사용해야 한다

id는 페이지내에서 고유한 값을 가지기 때문에 class를 사용하는 것이 맞다

찾을 때도 #main-menu .first 이렇게 찾으면 된다

좀 더 명확하게 하기 위해서는

#main-menu li.first라고 하면 li에서 first를 찾아라

좀 더 더 명확하게 하기 위해서는 (밑에 다른 ul에 li가 first일 수 있으니까)

#main-menu>ul>li.first(자식의 자식의 자식에서 first를 가지는 것을 찾아라)

#main-menu>ul>li.first+.aa라고 하면 first태그 바로 밑에 aa라는 이름을 가지는 태그를 찾는다

#main-menu>ul>li.first ~.aa라고 하면 first이후에 aa라는 이름을 가지느 태그를 찾아서 적용한다

===========================================

**선택자 우선순위**

적용범위가 넒은 selectors 일수록 우선순위가 낮다

id > class > 태그

동일한 연산 순위와 재정의

.h1{color : green;}

.h1{color : orange;}

.h1{color : red;}

-> 이런식으로 작성한다면 덮어쓰기가 된다(오버라이딩)

복합 연산자의 연산 순위

h1.h1 {color : green;}

section > .h1 {color : orange;} -> 콤비네이션 연산자를 사용한 것이 우선순위가 높다 (그 중에서도 class의 h1이니까 더 높다)

section > h1 { color : blueviolet;}

.h1{color : red;}

h1[lang="ko"]{color : blue;} -> 속성이 들어가 있기 때문에 이게 우선순위가 높다

.h1{color : red;}

h1{color : gray;}

===========================================

**스타일 링크하기**

-> 공통으로 사용하는 스타일을 모아서 style.css로 만든다 추상화시키는 느낌

-> 링크 태그는 반드시 헤드 태그 안쪽에 사용해야 한다

<link rel="stylesheet" href="../css/style.css" type="text/css">

- rel -> 관계를 나타내는 것

- type -> 링크하는 것의 타입은 텍스트형식의 css이다

===========================================

**스타일 리셋과 평준화**

-> 기존에 HTML5가 가지고 있던 스타일을 완전히 없애버리고 다시 작성하는 것

(reset.css: 완전히 리셋 / normalize.css : 리셋하고 조금만 수정해서 사용)

평준화를 위한 방법 1 : Normalize

평준화를 위한 방법 2 : Reset Css

구버전의 브라우저 지원하기

평준화를 위한 방법 3 : Modernize

(HTML5가 적용되지 않는 브라우저가 있을 수 있다)

(HTML5Shiv -> Check Enabled)

초기화 탬플릿

-> initializr.com

===========================================

**스타일 적용 순서**

li 태그는 ul과 ol태그 사이에 사용가능 하다 중간에 div나 이런 태그가 들어오면 안된다

또한, 테이블 태그에서도 table tr th(td)이런 순서로 가야지 중간에 li가 들어가거나 하면 안된다

컨텐츠 블록 스타일 -> 레이아웃 블록 스타일

(컨텐츠 블록을 스타일하면 컨텐츠 블록이 들어갈 방을 꾸며야 하는데 이걸 레이아웃 블록으로 꾸민다)

(컨텐츠 블록이 가구라면 가구를 배치할 단위 블록을 레이아웃 블록이라고 한다)

(레이아웃 블록 스타일은 방을 꾸미는 것이고 컨텐츠 블록 스타일을 가구를 꾸미는 것이다)

(일반적으로 방을 먼저 꾸미고 가구를 꾸민다)

레이아웃이란???

-> 배치하다 배치하기 위해서는 방이 필요하다

-> 과거에는 모든 배치를 table 태그로 해결 -> 지금은 css로 해결

격자형으로 보이는 방

-> 한방향으로 방을 여러번 중첩되는 방을 사용해야 한다

격자형은 단 방향을 중첩해서 만든다

-> 일자로 방을 나누고 그 안에서 또 방을 나누고 그 안에서 방을 나눠서 컨텐츠가 들어갈 방을 선택한다

웹 사이트의 기본 레이아웃 -> 방 나누기

-> 방을 잘못 나누면 배경을 이어갈 때 힘들다

===========================================

**제일 큰 방 설정**

block 레이아웃을 이용한 방 만들기

가장 큰 방을 행으로 나누어서 높이를 설정한다

header footer는 문서 전체에서 한번만 올 수 있는게 아니다 (영역이기 때문이다)

하지만 main은 문서 전체에서 하나만 올 수 있다

즉, main을 제외하고는 한번씩 더 사용할 수 있는 태그이기 때문에 id로 선택자를 설정하는 것이 바람직 하다

이때 id는 태그의 이름으로 하는 것이 덜 헷갈리는 방법이다

박스의 높이를 설정해주지 않으면 박스의 높이는 컨텐츠의 높이만큼 설정된다

박스가 줄어들었다고 컨텐츠가 줄어들 수 없으니까

-> 그리고 색을 입혀보면 테두리가 있는 것을 확인할 수 있다. 이게 기본 HTML5의 스타일이다

-> 이 테두리를 없애주려면 reset을 해줘야 한다

css에서 설정을 무효화 시키려면 앞에 --을 붙여주면 된다

live server로 열어서 개발자 도구로 들어가서 테두리를 확인한다

옆에 태그와 거리를 두는 것이 margin

body에 margin : 8px라는 값이 default로 지정되어 있다

그래서 reset.css파일을 만들어서 body의 값을 0으로 덮어쓰기 할 것이다 (오버라이딩)

여기서 reset.css를 링크하는 방법

1. <link ref="stylesheet" type="text/css" href="../css/reset.css">

2. style.css 파일에 가서

@import url(reset.css);

-> import하면 단점 웹 브라우저에 로딩할 때

html을 읽어드리는데 link에 css을 읽고 css에 들어가서 import되어 있는 것을 다시 reset.css가서 찾는 것이기 때문에 속도가 느릴 수 있다

-> 그래서 요청시간이 길어질 수 있다

-> 사용할 때 요청시간이 길어지면 1번 방법을 사용하고 아니면 2번을 사용하는 것이 낫다

===========================================

**color 값**

Color Keywords vs. Hex vs. Decimal

RGB hex values

#c0(red)c0(grenn)c0(blue) -> 24비트로 색상을 표현한다

맥시멈 값이 그래서 f가 된는 것이다 (#ff0033 -> #f03으로 줄여서 사용할 수 있다)

RGB decimal values

rgb(255, 0, 0);

-> 숫자와 퍼센트 둘 다 사용가능

-> but ! 섞어서 사용하는 것은 안됨 / 소수점 안됨

Transparent

Alpha Channel

rgb(255, 0, 0); - alpha channel -> rgba(255, 0, 0, 0.2);

-> 0~1까지 나타낼 수 있으며 a는 투명도를 말한다

-> 0에 가까울수록 투명해진다

-> #rgba => #ff000020; 이렇게도 사용가능

HSL(Hue-saturation-lightness model)

hsl(0, 100% 50%) red

hsl(120,100%50%) green

system colors

ActiveBorder - Active window border

===========================================

**두번째 방 설정하기 (박스 정렬과 최소 높이)**

컨텐츠 영역이 보이는가?

header안에 div를 줘서 class이름을 content-box라고 한다 그리고 찾기 쉽게 css파일에서는

header 밑에 작성한다 #header>.content-box

#header>.content-box{

height: 70px;

background: navy;

}

-> 이렇게 사용할 경우 높이가 변경되면 계속 같이 바꿔줘야 한다.

-> 그래서 사용하는 방법은 height : inherit라고 해서 부모의 것을 상속받겠다

-> 두번째 방법은 height : 100%라고 해도 된다

-> 그럼 두개의 차이점은?

-> 박스의 크기를 줄여도 상대적으로 크기를 동일하게 하고 싶으면 ??

블럭 스타일의 정렬 방법

-> width로 폭을 조절한다

-> widht는 부모의 너비를 다 차지한다

.content-box{

width:960px; -> 이렇게 주면 그대로 왼쪽 정렬이 되어버린다

margin-left:auto; -> 왼쪽에 자동으로 마진을 주겠다 / 그러면서 오른쪽 정렬이 된다

margin-right:autp; -> 이렇게 하면 오른쪽에도 자동으로 마진을 주겠다 / 그러면 가운데 정렬이 된다

}

-> 즉, 박스를 가운데 정렬하고 싶으면 margin-left, margin-right를 auto로 지정해주면 된다

이제 여기서 header말고도 다른 영역에서도 동일하게 가운데 정렬을 하고 싶다

===========================================

**세번째 방 설정**

section은 제목을 가지고 있는 영역(문서의 제목을 감싸는 블럭) -> 이것이 스타일을 이야기하는 것은 아니다

방크기 설정

aside 와 main의 방 크기 설정하기

블럭 박스의 레이아웃 방식

블럭(block)형식의 기본 배치 방식 : static형식의 포지션을 가진다 /

-> 배치 정보를 지정하지 않으면 블럭이 겹치지 않으면서 다음칸으로 넘어간다

블럭1

블럭2

블럭3

이렇게 있을 때 블럭2의 크기를 줄였다고 블럭3이 블럭2의 줄인 공간으로 올라갈까?? NO

블럭을 상하형식이 아닌 좌우형식으로 쌓고 싶을 때 : FlOAT를 left로 변경하면 좌우로 쌓인다

하지만 지금은

display : flex;라는 옵션으로 박스를 수평으로 옮기거나 수평으로 정렬할 때 사용된다

콘텐츠 레이아웃과 절대위치

앱을 닮아가는 웹의 레이아웃 기법

레이아웃 : 아이템을 배치하는 것

WAHAWG에서 웹을 플랫폼으로 보면서 어플리케이션에서 적용되던 레이아웃 기법이 웹에도 적용되는 것이다

이제는 레이아웃 기법이 포함되어서 flex가 가능해진 것이다 (Grid도 지원하기 시작함)

콘텐츠 블록의 일부분의 위치를 정하기

(Absolute, Relative, fixed, sticky)

#s1 ul {

background: hotpink; /\*색이 안보인다 why? 자식이 부모의 영역을 다 차지하고 있어서\*/

height: 300px;

position: relative; /\*position을 relative로 주게 되면 li은 ul영역 안에서만 움직인다\*/

}

#s1 li:nth-child(2){/\* nth-child(2) : li의 두번째에 적용하겠다\*/

background: green;

position: absolute; /\* position을 absolute로 지정해주면 내가 지정한대로 움직여

이렇게 되면 li 2번째는 li에서 빠지게 되고 폭과 넓이는 콘텐츠의 크기만큼

지정이 된다

absolute를 하게되면 형제들끼리 겹치게 되는데 밑에 깔리는 것이 아니라

위로 올라오게 된다\*/

left: 100px; /\*오른쪽으로 붙이고 싶으면 right를 0px로 준다\*/

top: 100px; /\*부모가 relative가 되면 그것이 기준이 되어 움직인다\*/

}

===========================================

**상대 위치** (position : relative)#s1 li:nth-child(1){

/\* nth-child(1) : li의 첫번째에 적용하겠다\*/

background: red;

position: relative; /\*원래 위치에서 50왼쪽 30위쪽으로 이동한 것이다\*/

left: 50px;

top: 30px;

}

===========================================

**고정위치**(position:fixed)

#s1 li:nth-child(3){

background: blue;

position: fixed; /\*absolute는 문서를 기준으로 따라가는 것이고

fixed는 브라우저의 화면영역에 고정되는 것이다

그래서 스크롤하면 같이 따라오는 것처럼 보인다

도구바 같은 경우 사용된다 따라오는 거

유튜브 상단에 검색창 고정된거 fixed 사용한거네

모니터에 스티커 붙여놓고 고정시키는거랑 비슷하다\*/

left: 100px;

top: 100px;

}

===========================================

**붙임 위치**(position : sticky)

#header{

height : 70px;

background : black;

position : sticky;

width : 100%;

top : 0px; -> sticky를 할때는 top을 0px로 줘야지 상단에 고정이 되고 스크롤 할때 같이 내려온다

}

Flexible Box Model and Terminology

문서를 위한 레이아웃

기존 CSS2의 레이아웃을 위한 Display 방식

display : (block, inline - 옆으로 배치, inline-block - 옆으로 블럭배치, list-item, table, flex, grid)

방향 설정

flex-direction : row | row-reverse | column | column-reverse;

래핑 설정

flex-wrap : nowrap | wrap | wrap-reverse

메인축 정렬 옵션

justify-content : flex-start | flex-end | center | space-between | space-around;

flex-start

123

flext-end

123

flex-center

123

space-between

123

space-around

1 2 3

수직축 정렬 옵션

align-item : stretch | flex-start | flex-end | center | space-between | space-around

stretch

숫자 길게

123

flex-start(수직 시작점)

123

flex-end(수직 끝점)

123

center

123

space-between

123

456

space-around

123

456

===========================================

**vscode 설정하기**

ctrl + , ( 설정 )

zoom : 확대 축소

theme : 색상 테마

asso : 확장자 연결 (\*.html html 설정)

확장에서 live server 검색 - 설치

확장 설정 아래쪽 127.0.0.1 값을 자신의 ip주소로 변경

이후 파일 편집중 우측 하단의 Go live를 클릭하면 브라우저에서 실행

(방화벽 허용, 연결 프로그램 : Chrome, Edge..../src/img.gif)

ctrl + , : 설정

ctrl + b : 좌측 탐색기 숨김/열기

ctrl + ` : 터미널 열기

F11 : 전체 화면

alt + shift + ↓ : 밑으로 복사

<!-- 자주 사용하는 HTML 태그 -->

https://www.advancedwebranking.com/html/#doctype

h1 : 한 페이지에 한번만 사용하는 것이 규약 ( headline level1 )

===========================================

**절대경로, 상대경로**

href, src 와 같이 특정 자원을 가리키는 속성에는 url을 작성하는데

if, http:// https:// ftp:// 등과 같이 프로토콜로 시작하면 절대경로

(대한민국 부산시 해운대구 우2동 센텀드림월드 1104호)

url이 protocol 없이 문서 이름이나 간단한 폴더 및 문서 이름만 작성하게 되면 상대경로

(1104호 어느 건물에서 가리키느냐에 따라 서로 다른 대상을 찾게 된다)

id로 찾는 것은 현재 문서 안에서 찾는 것이다

해당 웹 문서의 id를 알고 있으면 url에서 바로 찾아갈 수 있다

===========================================

===========================================

**padding, margin**

padding : 테두리 기준 안쪽 여백

margin : 테두리 기준 바깥쪽 여백

상하좌우 값을 모두 줄 수 있다

padding에 의해 전체 크기가 바뀐 것을 확인할 수 있다

값을 두번 주면 상하좌우값을 줄 수 있다

--> padding은 내부 margin은 외부

===========================================

**flex**

flex 내부를 정렬하겠다

flex-flow : wrap -> 가로로 배치하겠다

<body style="display: flex; flex-flow: wrap;">

display

- flex로 지정한 요소는 자신이 아닌, 내부 요소를 기준에 따라서 정렬하는 요소이다

- main axis(기본 축) 와 cross axis(교차 축)에 따라 정렬을 수행한다

- justify-content 속성으로 main축 기준 정렬 위치를 지정할 수 있다

flex-start : 축의 시작지점쪽으로 요소를 배치한다

flex-end : 축의 끝점 쪽으로 요소를 배치한다

center : 축의 가운데를 기준으로 배치한다

space-between : 축의 시작점과 끝점에 양긑 요소를 배치하고 간격을 두며 배치한다

flex-flow 속성으로 축을 교차시키거나, 범위를 벗어날 경우 처리를 변경할 수 있다

flex-flow 기본값을 row이다

flex-flow : row default값

flex-flow : column main축이 위에서 아래로 내려오는 방향으로 바뀐다

flex-flow : wrap 범위를 벗어나면 자동으로 다음 위치로 변경된다

flex-flow : column wrap-reverse 오른쪽 위에서부터 채운다

===========================================

1px : 선두께 solid : 선 유형 (solid : 선, dashed : 점선)

margin : 테두리 바깥 쪽의 경계의 값

<div style="border: 1px dashed black; width: 100px; height: 100px; margin: 10px;"></div>

==============================================

**css 선택자 (selectors)**

1) 태그 선택자 : 지정한 이름의 모든 태그에게 속성을 적용한다

body {}

2) id 선택자 : 지정한 id를 가지는 문서의 첫번째 요소(id는 문서에서 고유해야 한다)에 속성을 적용

#id{}

3) class 선택자 : 지정한 클래스를 가지는 문서 내의 모든 요소에 속성을 적용한다

.class {}

4) 자손 선택자 : 지정한 요소 바로 아래쪽에 속한 요소들에게 속성을 적용한다

--ul li {}

ul>li {}

--ul+li {} : 인접한 li에게

--ul~li {} : ul밑에 있는 li

5) 후손 선택자 : 띄워쓰기로 구분하고, 지정한 요소 아래의 모든 요소(단꼐가 멀어도 상관없음)에 속성을 적용

ul li {}

===========================================

**알고 있어야 하는 CSS 속성**

[테두리 관련]

border: [두께][유형][색상]

border-radius: px, %

width: px, em, rem, %

height: px, em, rem, %

padding: px, em

padding-top

padding-bottom

padding-left

padding-right

margin: px, em, auto(지정된 div안에서 가운데 정렬해준다/ 바깥쪽 여백을 알아서 맞춰라)

margin-top

margin-bottom

margin-left: auto(오른쪽 정렬)

margin-right: auto(왼쪽 정렬) -> 두개 다 사용하면 가운데 정렬

[배경 관련]

background-color: [색상이름], [RGB 코드], rgba, hsl, rgb(255,255,255);

background-image: url();

background-size: 100%

background-position: center;

background-repeat: no-repeat

overflow: hidden (영역을 벗어나는 컨텐츠를 자동으로 숨겨준다)

sroll, auto

[글자 관련]

color: px, pt, em

font-size: 글꼴, 클라이언트 컴퓨터에서 읽을 수 있는 글꼴

font-family: bolder(굵게)

font-weight: px, pt, em

font-height: px, pt, em

line-height: 한줄이 해당 px을 차지하는 것 / 영역의 높이와 글자 한 줄의 높이를 맞추고 싶을 때

box-sizing: border-box; /\* 테두리를 포함하여 전체 크기에 맞도록 계산한다

fonts -> 실행창에 검색하면 글꼴을 검색할 수 있다

===============================================

**반응형 웹 만들기**

viewport 사용법

반응형 웹 (responsive web)이란 하나의 웹사이트로

데스크탑 PC, 스마트폰, 태블릿 PC 등 접속하는 디스플레이의 종류에 따라

화면의 크기가 자동으로 변하도록 만든 웹 페이지를 말합니다.

미디어 쿼리는 CSS3 부터 지원이 되는 CSS기술로 미디어 타입, 화면 크기 등을 기준으로 다른 스타일 시트를 적용할 수 있도록 해줍니다.

이를 이용해서 화면 크기가 변할때 스타일을 바뀌도록 해서 반응형 웹을 구현할 수 있습니다.

모바일 웹을 작성하게 되면 뷰포트(viewport)라는 것을 고려하게 됩니다.

뷰포트는 웹페이지가 사용자에게 보여지는 영역을 말합니다.

데스크탑 PC에서 브라우저의 크기를 줄이면 웹페이지의 내용이 다보여지지 않고 스크롤 해서 봐야 되는 경우가 있습니다.

이때도 브라우저에 보여지는 부분이 뷰포트입니다. 데스크탑 PC는 브라우저의 크기를 바꿔서 뷰포트의 크기를 바꿀 수 있는 것입니다.

반면에 휴대폰이나 태블릿의 경우는 브라우저의 크기를 변경할 수 가 없습니다.(요즘은 멀티뷰를 지원하는 기기도 있지만, 일반적이지는 않습니다.)

하지만, 워낙 다양한 기기들이 존재하기 때문에, 뷰포트의 크기도 다양합니다.

휴대폰이나 태블릿이 나오기 이전의 웹페이지는 데스크탑 PC용으로만 만들었기 때문에 고정된 크기를 가지는 것이 대부분이 었습니다.

이렇게 고정된 크기를 가진 웹페이지를 화면이 작은 휴대폰에서 보면 한 화면에 다보이지 않게 됩니다.

대부분의 휴대폰등 모바일 기기의 웹브라우저에서 웹페이지가 그려지는 렌더링 영역은 사용자에게 보여지는 영역인 뷰포트 보다 큽니다.

1024 x 768 사이즈에 고정되어 있는 웹페이지를 320 x 568 사이즈를 가지는 아이폰에서 보면 스크롤해야만 모두를 볼 수가 있습니다.

HTML5에서 소개된 뷰포트 <meta> 태그를 사용하면 모바일 기기에서 실제 렌더링되는 영역과 뷰포트의 크기를 조절할 수 있습니다. 또한 줌 레벨도 조정할 수 있습니다. 아래는 가장 일반적으로 사용되는 설정 입니다.

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

- width=device-width : 페이지의 너비를 기기의 스크린 너비로 설정합니다. 즉, 렌더링 영역을 기기의 뷰포트의 크기와 같게 만들어 줍니다.

- initial-scale=1.0 : 처음 페이지 로딩시 확대/축소가 되지 않은 원래 크기를 사용하도록 합니다. 0~10 사이의 값을 가집니다.

이것외에도 다음과 같은 값을 지정할 수 있습니다.

- minimum-scale : 줄일 수 있는 최소 크기를 지정합니다. 0~10 사이의 값을 가집니다.

- maximum-scale : 늘릴 수 있는 최대 크기를 지정합니다. 0~10 사이의 값을 가집니다.

- user-scalable : yes 또는 no 값을 가지며 사용자가 화면을 확대/축소 할 수 있는지는 지정합니다.

위의 줌 레벨은 1이 원래크기이고, 0.5 라면 50% 축소를 뜻합니다.

그러므로 위의 설정은 페이지 너비를 휴대기기 화면너비에 맞추고, 처음 로딩시 줌을 하지 않는 것입니다. 적용한 결과는 다음과 같습니다.

이미지에 대해서는 모바일에서 사용하기 위해서는 좀 더 처리가 필요합니다.

화면이 클 때는 자신의 크기를 유지하고 화면이 이미지의 크기보다 작아지면 화면크기에 맞춰서 크기가 줄어들도록 CSS를 지정할 수 있습니다.