**<Module #1 : The tools of a Web Developer>**

**\* 텍스트 에디터**

->웹사이트 = 텍스트.

->텍스트 에디터는 이러한 텍스트 편집을 편리하게 만들어주는 도구이다.

->atom, VSC(Visual Studio Code)등이 존재.

**\* VSC(Visual Studio Code)**

->추천하는 텍스트 에디터.

->Extension(추가 기능)을 이용하여 해당 에디터를 더 편리하게 사용할 수 있다.

->초보자는 Prettier, Materail Theme과 같이 코드를 자동으로 정리해주는 Extension정도만 사용하는 것을 추천.

**\* Git & GitHub(버전 관리, 백업, 협업)**

->Git : 코드 변경 사항을 추적하는 시스템(버전 관리를 쉽게 해주는 시스템)(Local).

->GitHub : 이러한 깃 파일을 올리는 인터넷 클라우드(백업을 쉽게 해주는 시스템)(Server).

(1) Repository : 코드 저장소.

(2) commit : 작업을 변경했다는 것을 간결하고 효율적으로 알림.

(3) Push & Pull : 작업 내용을 Server 로 보내거나(Push), Local 로 가져옴(Pull).

(4) branch

->master branch : 모든 작업의 기록, 다른 사람들에게 공개 되고 참조됨.

->branch : 메인 작업창 이외의 실험을 해볼 수 있는 다른 작업의 기록.

**\* Html(hyper-text-markup-langauge)**

->hyper text : 비순차적으로 기억된 텍스트 데이터.

->내가 작성한 텍스트의 어디 부터 어디 까지가 무엇을 의미하는지 마킹 해주는 도구.

(Ex : 여기서 부터 여기까지 타이틀, 링크, 리스트.. 이다 를 html로 알려주는 것).

**\* CSS(cascading-style-sheet)**

->cascade : 작은 폭포, 직렬

->브라우저에게 각각 요소들이 어떻게 생겨야 하는지 알려주는 도구

(Ex : html 이 마킹한 부분의 색, 이미지의 크기 등등을 CSS가 설정해주는 것)

**<Module #2 : HTML5>**

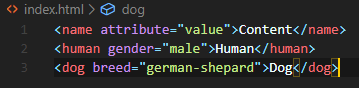
**\*html(hyper-text-markup-language)**

->웹사이트에서 각 요소가 무엇을 뜻하는지 알려주는 언어. 이를 tag를 이용하여 작성한다.

->html문서는 tag가 매우 많은 텍스트 문서일 뿐이다.

**\*Anatomy of a HTML tag**

->태그는 <이름 속성=”값”>내용</이름> 과 같은 모양으로 생겼다.



->링크 태그 : 구글 링크로 이동하는 태그 예.

<a href="http://google.com">Go to google</a>

->링크 태그2 : 구글 링크로 이동하되, 새 창으로 열기.

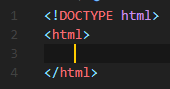


->href : hypertext reference의 줄임말.

**\*<!DOCTYPE html>**

->구글 크롬으로 하여금 이 문서가 html 문서임을 알려준다.

-><html></html> 태그로 문서를 열고 닫을 수 있다.



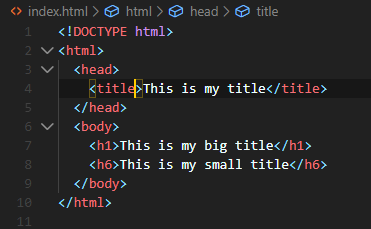
-><!DOCTYPE html>은 self-contatined tag라고 한다. 이 tag는 그 자체로 정보를 제공하기 때문에 혼자 스스로 열고 닫는 tag이다.

**\*html Document**

->문서는 크게 2가지 부분, head와 body로 나뉜다.

(1) head는 유저에게 보이지 않는 부분으로써, 브라우저에게 해당 웹사이트에 관한 필요한 정보를 제공한다.

(2) body는 사람들이 읽는 컨텐츠로써, 웹 사이트에서 보이는 부분이다.



-><h1></h1> tag는 **타이틀의** 글자의 크기를 조정할 수 있다. h1이 가장 크고, h6가 가장 작다.

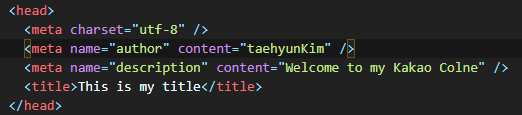
->tag는 항상 열고 닫아야만 한다.

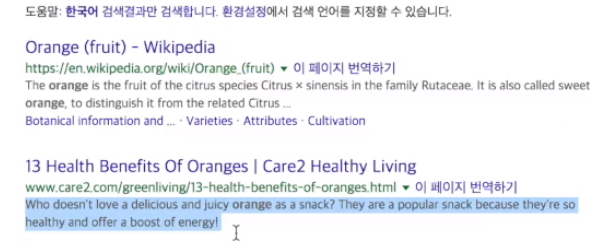
**\*Meta tag**

->meta : 추가 정보, extra.



->meta charset tag : 브라우저에게 현재 작성하는 문서가 어떤 character(언어)로 작성되는지 알려주는 tag. 인코딩 방식을 설정한다.



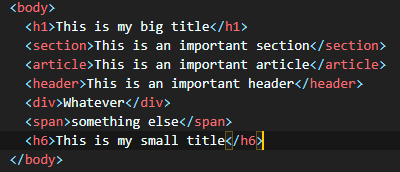


->위 그림과 같이 웹사이트를 간략히 설명하는 글들이 meta 내용이다. meta tag는 매우 여러가지 종류가 있으며, 어떤 태그는 특정 플랫폼만을 위해서 존재하기도 한다.

->즉, 이러한 정보는 브라우저를 위한 정보이기 때문에 head파트에 작성해야 한다.

**\*Semantic tag/Non-semantic tag**

->semantic : 의미가 있는 태그, non-semantic : 의미가 없는 태그.



->section, article, header와 같은 태그는 의미가 있는 태그이다. 즉, semantic 태그는 제목/문단/네비게이션 등등 뭔가 뜻이 있는 태그를 말한다.

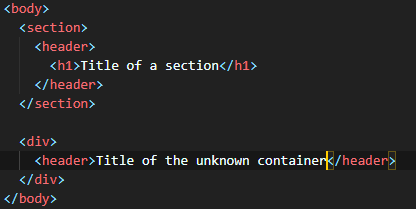
->div, span은 의미가 없는 태그이다.

div 태그는 컨테이너, 박스와 같은 태그이다. 안에 내용물을 넣는 용도이다.

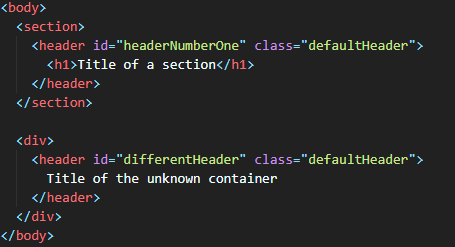
span 태그는 텍스트를 위한 컨테이너이다. title, 문단 p와 같은 태그와는 다른 용도로 쓰인다.

**\*Giving our tags a name(ID’s and Classes)**

->수 많은 태그를 이용하면 문서의 의미를 헷갈릴 수 있다. 예를들어 div 태그가 많으면 정확히 해당 부분이 무엇을 의미하는지 알 수 없다. 따라서 각 div가 무엇을 뜻하는지 알려줄 필요가 있다.



예를 들어 다음과 같은 상황이면 <section>태그의 헤더와 <div>태그의 헤더를 구분하기 어렵다.



->이는 ID와 Class를 이용하여 구분을 지을 수 있다.

ID는 각 element마다 고유의 값을 가지는 값을 의미하고,

Class는 서로 공유할 수 있는 속성을 의미한다.

위의 예시에서 두 부분은 ID로 구분되어 있고, 동일한 속성을 공유하고 있음을 알 수 있다.

->웹사이트 상에 아예 고유한 element는 많이 없기 때문에 Class를 많이 사용한다.

예를 들어 채팅창/메뉴바와 같은 것은 아이콘이 많고 계속 반복되기 때문에 Class를 주로 이용하게 되지만, 헤더/네비게이션등은 고유하기 때문에 ID를 이용하게 된다.

**<Module #3 : CSS3>**

**\*CSS Stntax**

->CSS는 .css를 확장자를 가지고 있다.

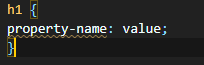
->CSS는 selector 파트와 property파트로 구분된다.

->property 파트



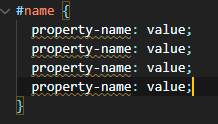
무조건 소문자, 중간에 공백이 없으며 값의 마지막 부분의 세미 콜론이 반드시 필요하다.

->selector 파트

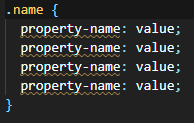


나의 html의 h1에 property를 연결하고 싶으면 다음과 같이 괄호로 묶어서 작성해준다.

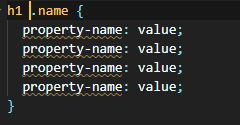
즉 여러 개의 property가 h1 파트에 적용될 것이다. 태그, 링크, 문단, 헤더, 푸터 등에도 모두 동일하게 적용된다.



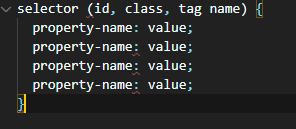
ID의 경우 다음과 같이 앞에 #을 붙여서 작성한다.



Class의 경우 다음과 같이 앞에 .을 붙여서 작성한다.



다음과 같이 태그와 클래스를 합쳐서 작성할수도 있다.

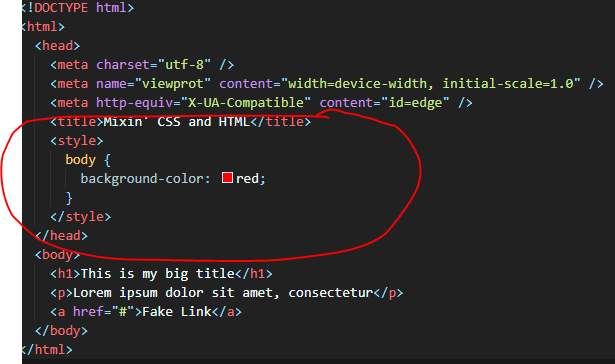


이런식으로 전체를 표현해줄 수도 있다.

**\*Mixing CSS wih HTML**

->HTML와 CSS를 연결하는 방법은 두 가지가 있다.

(1) inline한 방법

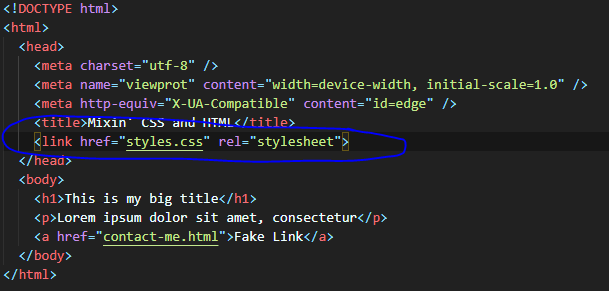


다음과 같이 HTML문서 안에 style 태그를 이용하여 직접 장성하는 것이다.

예를 들어 이는 여러 개의 웹 페이지가 존재할 때, 같은 배경색을 쓰고 싶은 경우 모든 HTML문서를 조작해주어야 하기 때문에 비효율적인 방법이다.

따라서 매 페이지를 만들 때마다 모든 CSS 변경사항이나 추가 사항을 일일히 복사-붙여넣기 해주어야 하기 때문에 이 방법은 추천하지 않는다.

(2) external한 방법



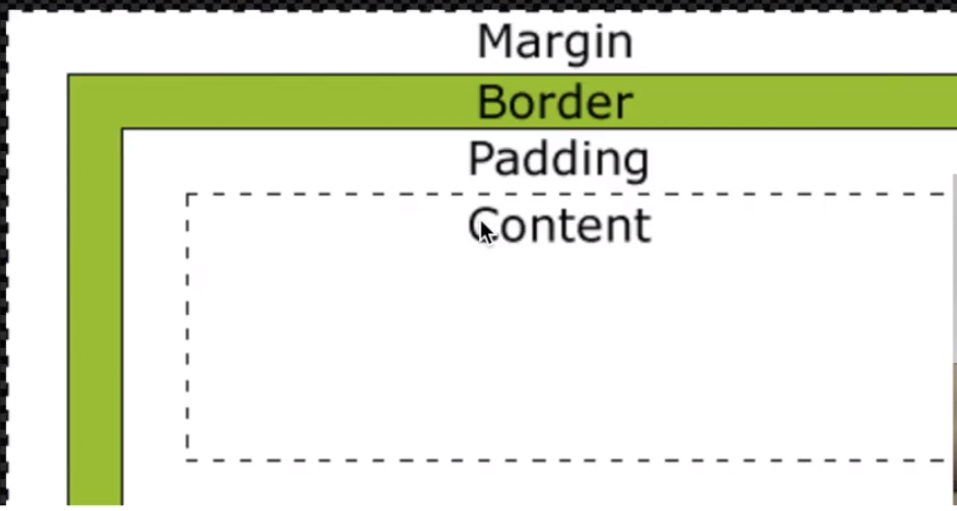
외부적인 방법을 이용하면 inline한 방법의 단점을 해결할 수 있다. 우선, 모든 HTML문서마다 다음과 같은 link를 이용하여 연결하고자 하는 CSS문서를 연결해 주어야 한다.



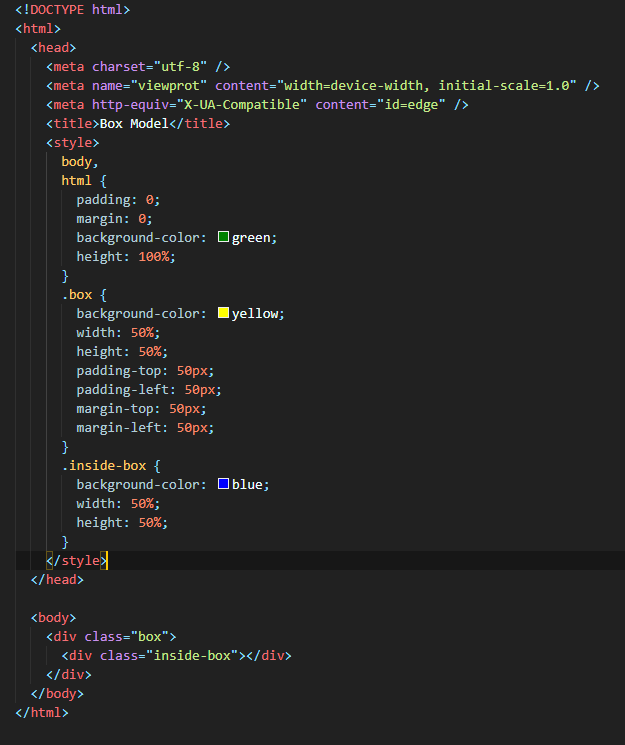
그리고 다음과 같이 원하는 CSS를 작성해 준다면 두 HTML 페이지 모두 CSS가 적용된 것을 확인할 수 있다.

**\*Box Model**

->CSS를 배울 때 이해해야 하는 것은, 많은 element들이 Box로 구성되어 있다는 사실이다.



박스는 첫번째로 컨텐츠(contents), 두번째는 박스의 보더(Border/경계), 세번째는 패딩(padding), 네번째는 마진(margin)으로 구성되어 있다. 패딩과 마진은 각각 박스와 보더 사이의 간격, 박스의 보더에서 바깥쪽으로 있는 간격을 의미한다.



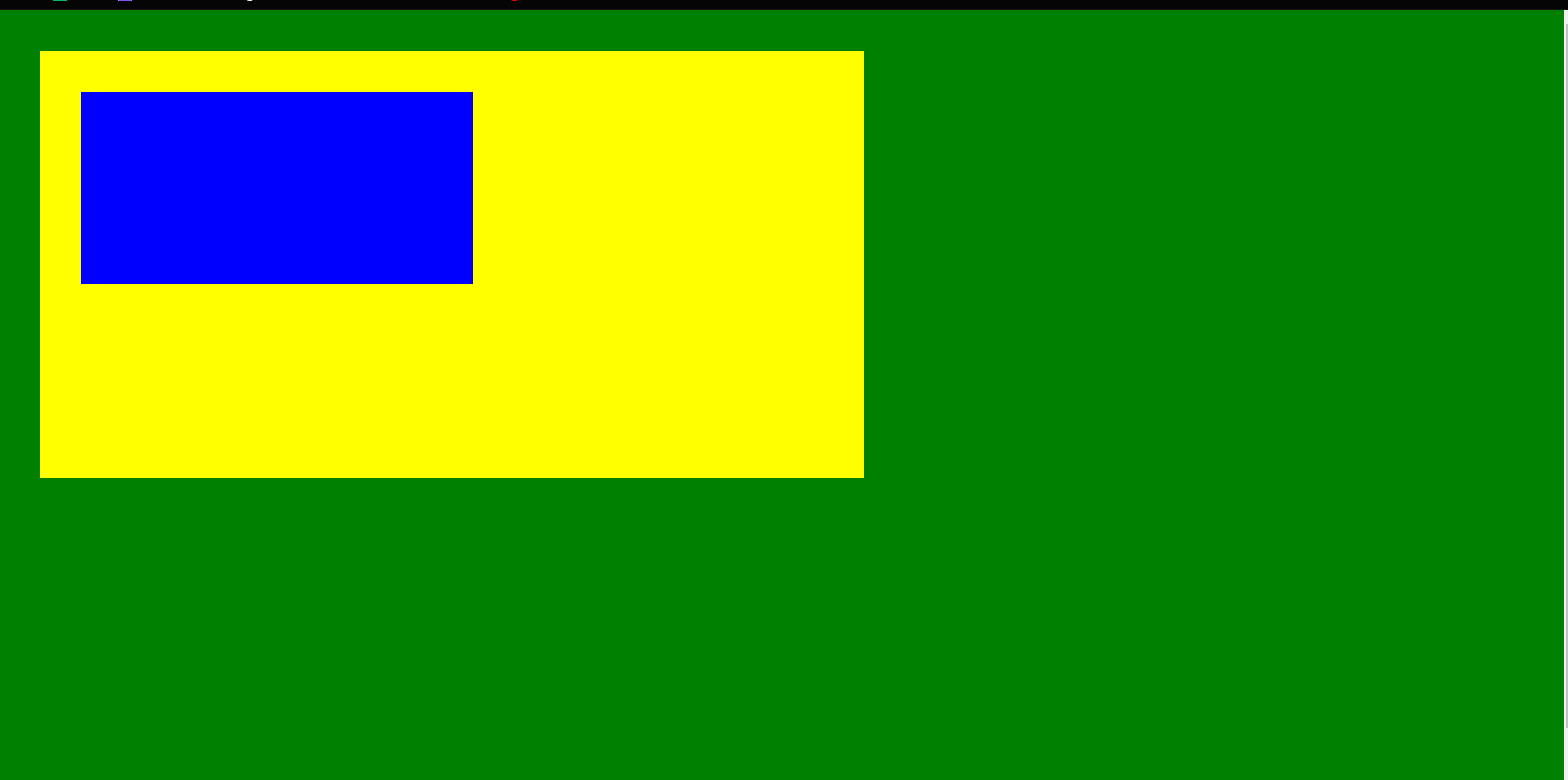
이와 같이 father-box에 padding과 margin만 설정해 주어도 아래 그림과 같은 웹 페이지가 형성이 된다.

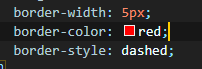
padding을 상하좌우 모두 주고 싶으면 padding : 값;과 같이 그대로 쓰면 된다.

padding을 상하/좌우 순서로 주고 싶으면 padding:상하값 좌우값; 과 같이 순차적으로 적어주면 된다.

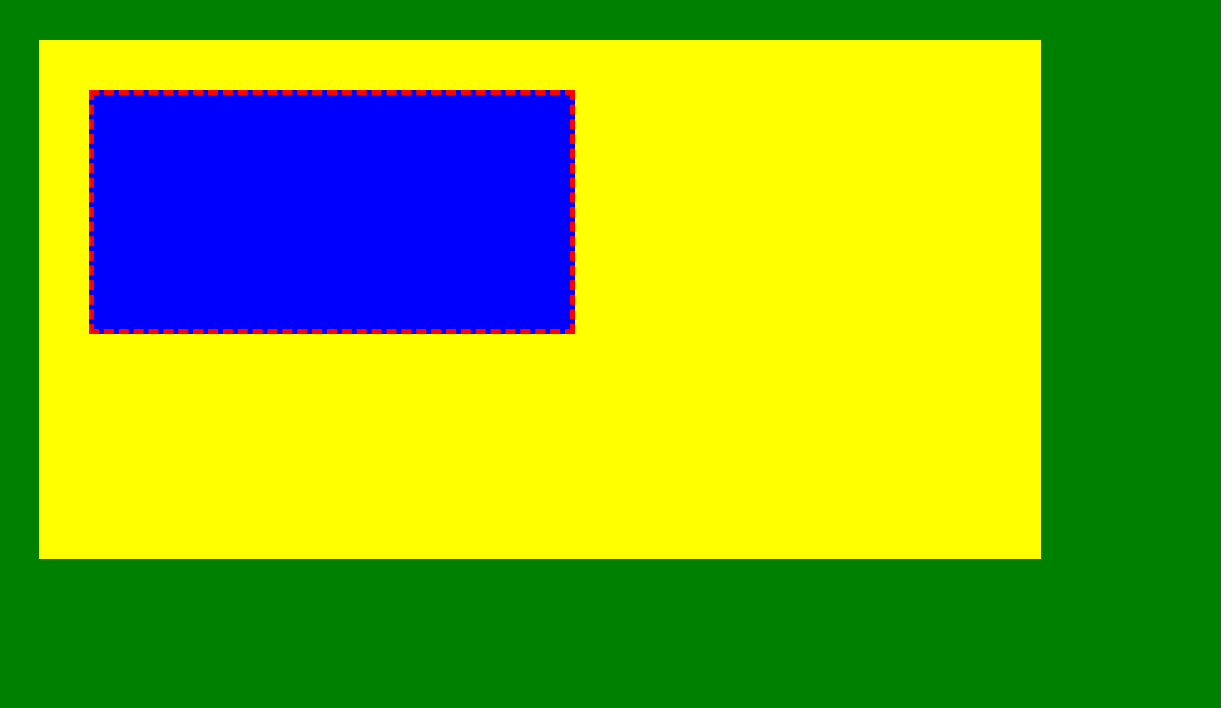
padding을 4방향 각각 다르게 주고 싶다면 시계 방향순으로 값을 적으면 된다(상-우-하-좌), 즉 padding:상 우 하 좌;

margin도 padding과 마찬가지 문법으로 작성할 수 있다.





보더-경계는 다음과 같은 3가지가 존재한다(넓이, 색상, 스타일).

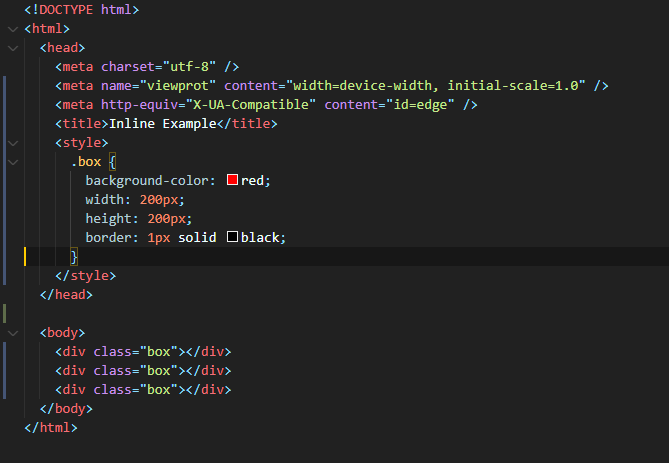


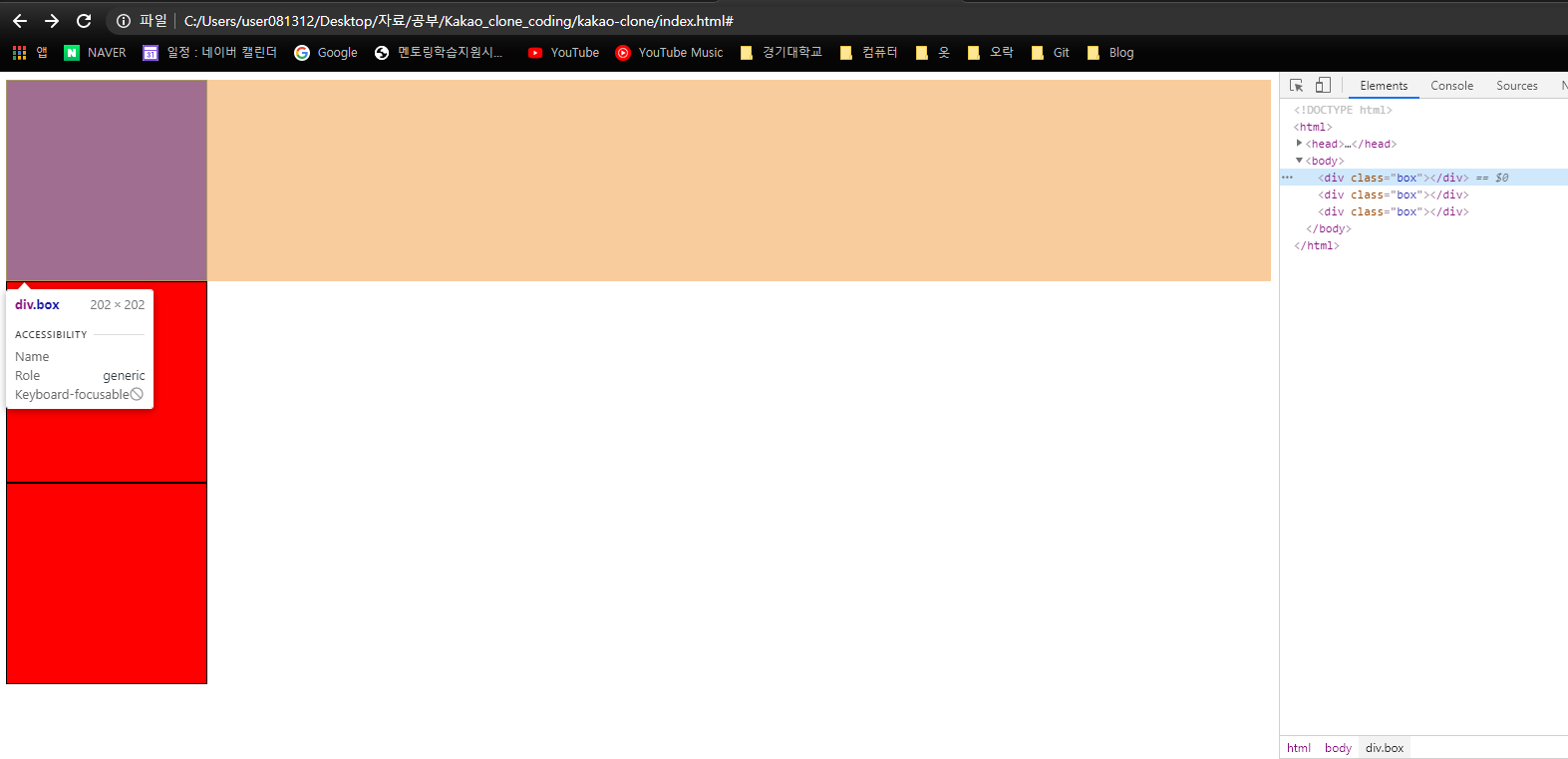


다음과 같이 줄여서 쓸수도 있다.

**\*Inline vs Block vs Inline Block**

->Block이란, 그 옆에 아무것도 존재하지 않는 경우를 말한다.

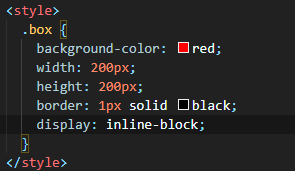




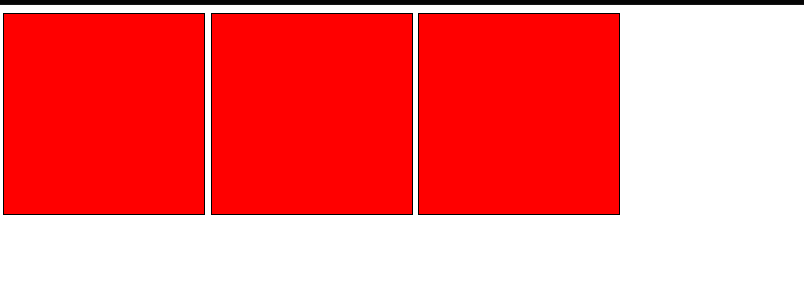
다음과 같이 페이지 검사를 수행해보면, 박스 옆의 요소는 아무것도 없는 것을 확인할 수 있다.

즉, 그 옆에 아무것도 존재하지 않는다는 것은, 무엇이는 그 밑으로 위치하게 된다는 것을 말한다.

->Inline Block

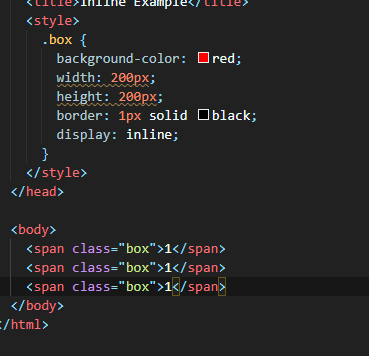


다음과 같이 display를 inline-block으로 설정해주면 다음과 같은 웹페이지가 생성된다.



인라인-블록은 박스들이 서로서로 바로 옆에 있을 수 있다. 디폴트 값은 블록 형태이다.

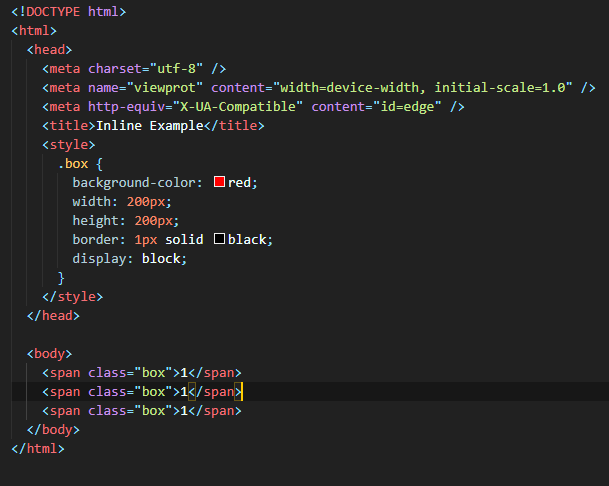
->Inline

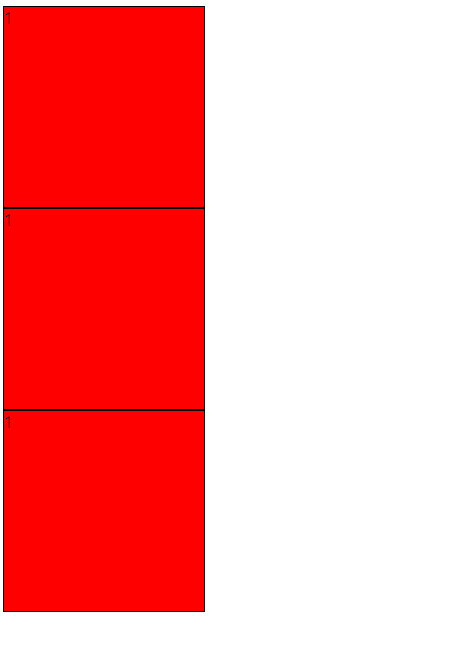




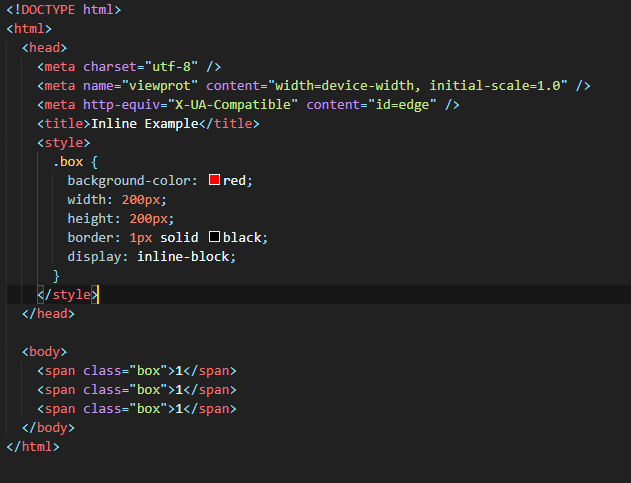
인라인 디스플레이는 텍스트 처럼 적용이 된다. 디스플레이를 인라인으로 설정하게 되면, 이는 블록이 아닌아닌 것 인식이 된다.

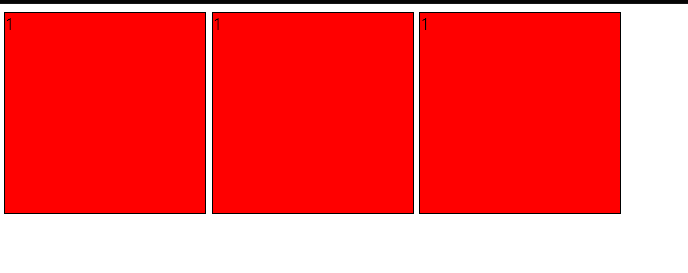
인라인 = 텍스트이다. 텍스트는 높이도, 폭도 없이 컨텐츠의 길이만 존재한다.

반대로 block은 폭, 높이가 존재하지만 그 옆에 어떤 요소도 놓을 수 없다.



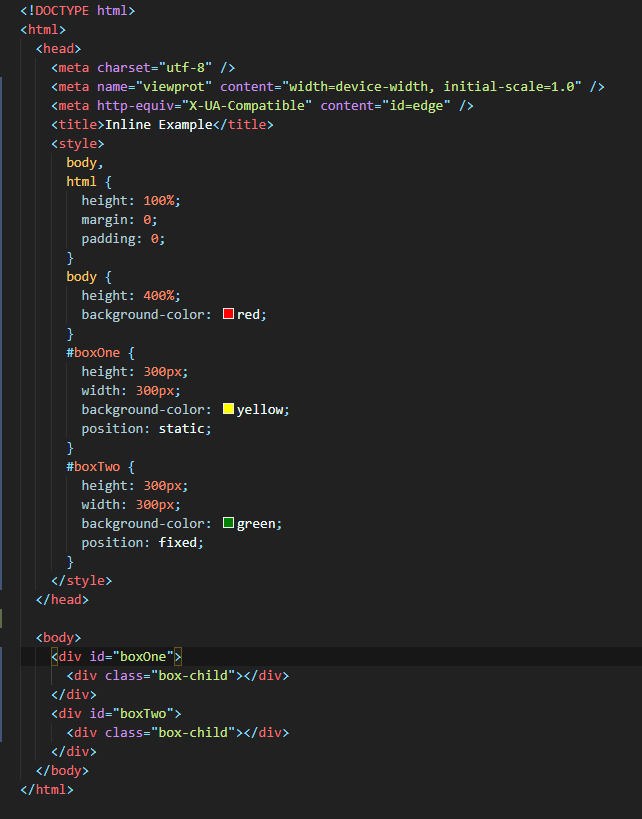
inline-block은 박스의 폭, 높이가 존재하고 그 옆에 서로서로를 배치할 수 있다.

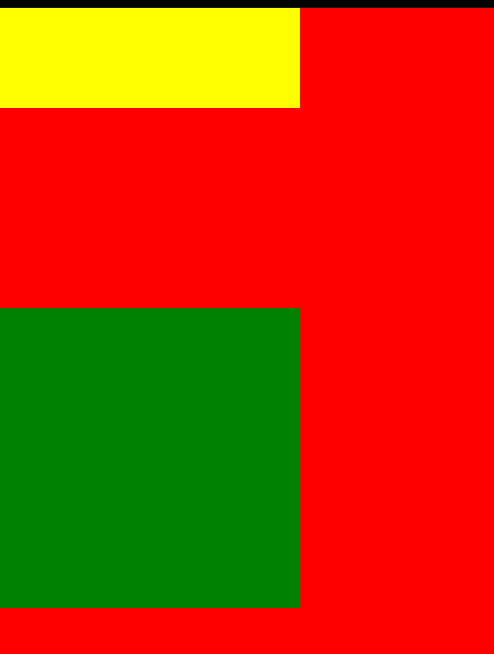
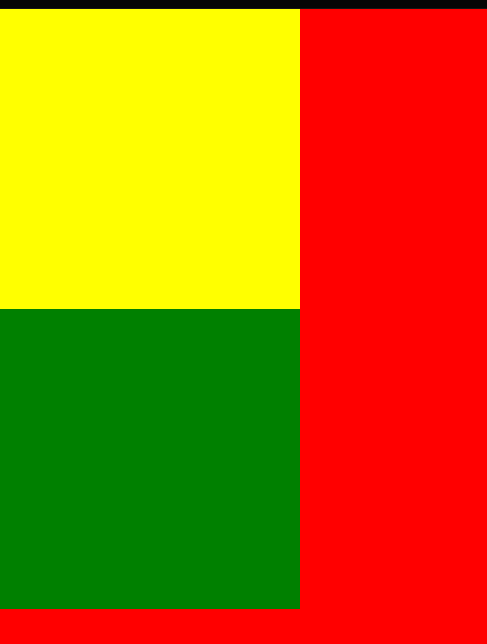




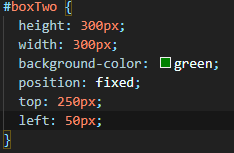
**\*Position property**

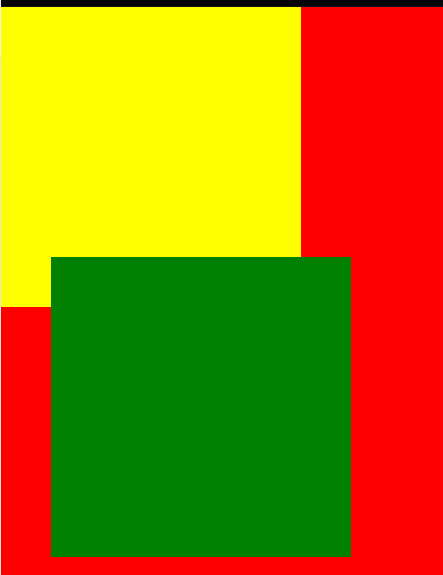
->static/ fixed position





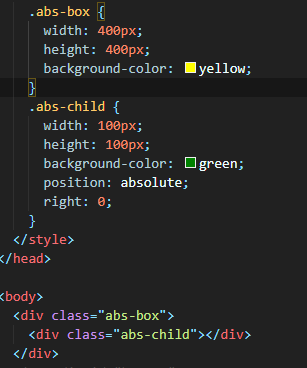
노란색 박스는 static 포지션, 초록색 박스는 fixed 포지션을 가지고 있다. static 포지션은 그 자리 그대로 고정이 되는 것이고, fixed 포지션은 디스플레이 상에서 상황이 변해도(스크롤 이동) 고정적으로 그 자리에 위치하게 된다. static 포지션은 디폴트 값이다.



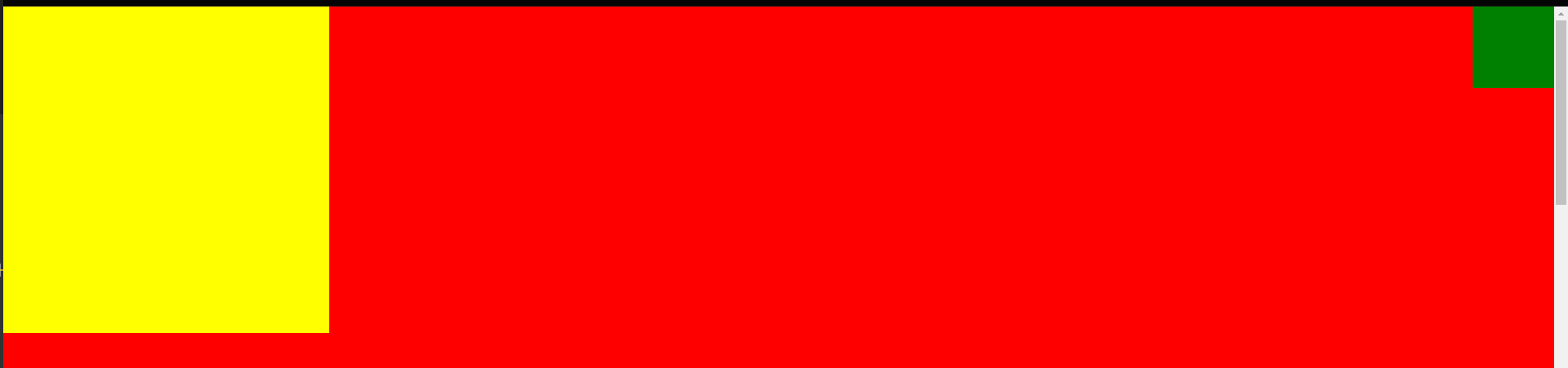


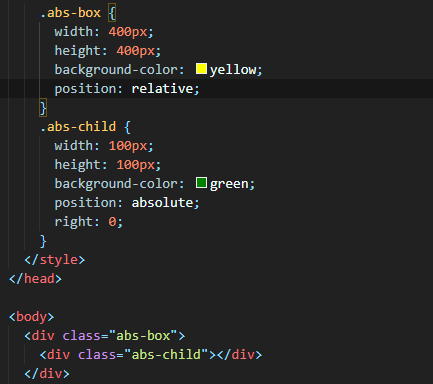
다음과 같이 포지션의 위치를 top, bottom, left, right를 이용하여 설정해 줄 수 있다.

->absolute/ relative position

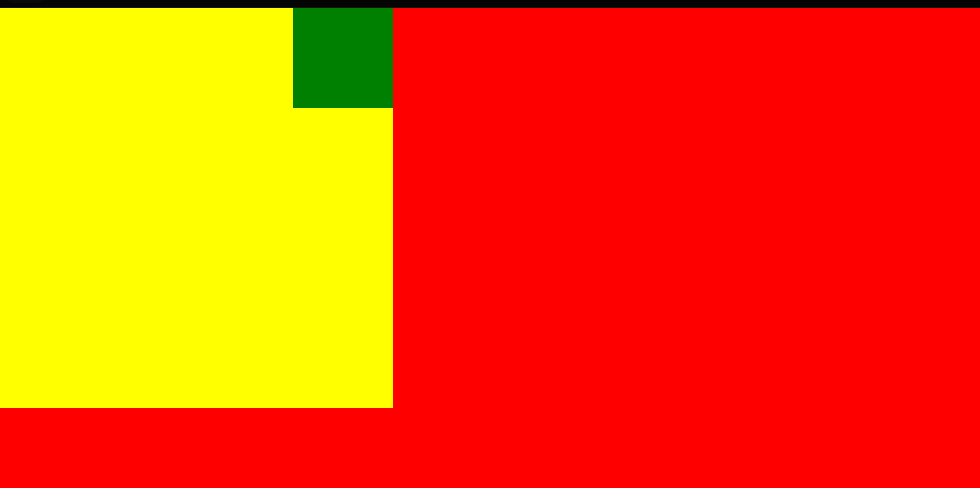


absolute 포지션은 html상에서 해당 element와 관계가 있는 element를 살펴보고, 이에 상응하여 포지션이 결정된다. 하지만 이 경우 부모 박스가 상응하는 관계가 없기 때문에 우리가 설정한 값대로 움직인다.



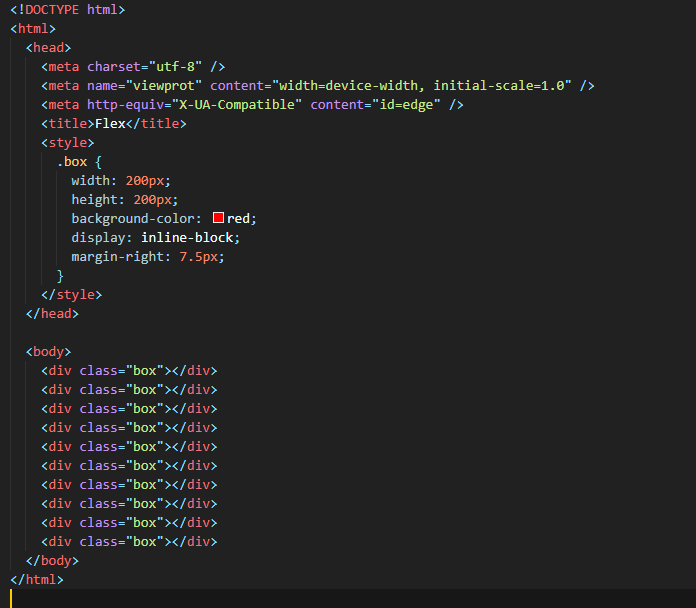


하지만 다음과 같이 부모 박스의 포지션을 relative로 설정해 준다면, 자식 box는 이를 인지하고 다음과 같은 출력을 보여준다.



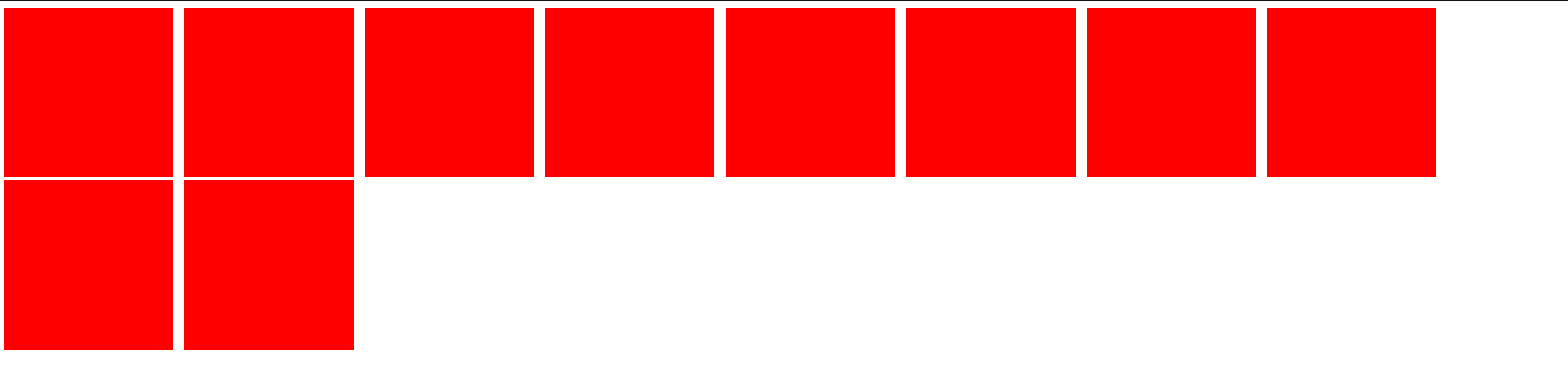
**\*Fluid layouts with Flexbox**

->flex가 필요한 이유



기존의 방법의 경우, box를 여러가 생성할 때 margin의 값에 따라서(위 6.5px, 아래 7.5px)만 해도 정형화된 디스플레이가 나타나지 않고, 원하는 디스플레이를 설정하기 위해 지루한 작업 과정을 거쳐야만 한다(즉, 자동으로 완성되는 그리드가 없는 문제점이 있다).





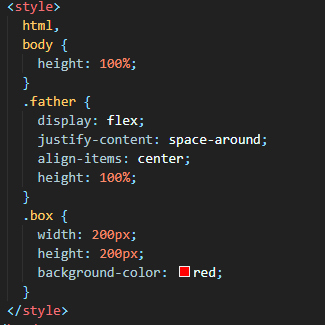
이러한 문제는 margin 조정 뿐 만 아니라 여러 부분에서 난해만 작업 사항이 될수도 있고, 출력되는 플랫폼(웹이아닌 아이폰)에 따라 다른 모습으로 출력되서 보이기 때문에 문제가 발생할 수 있다!

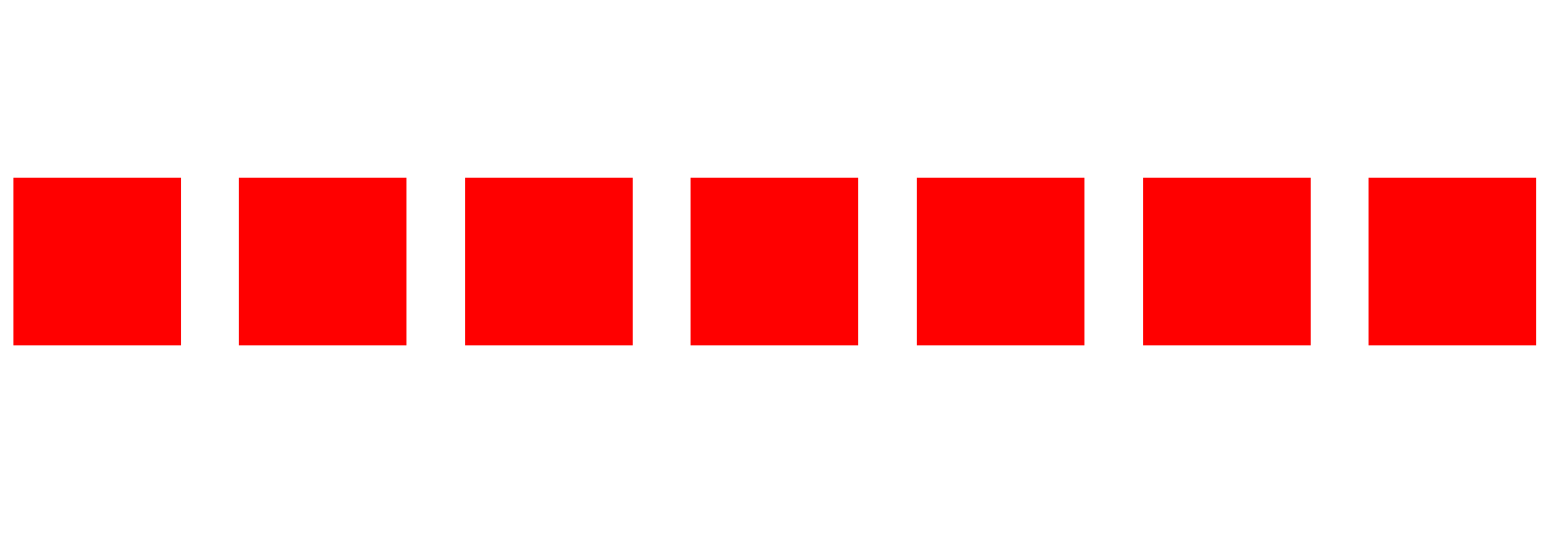
->flex의 이용

(1) 우선 부모 클래스를 만든다. 플랙스를 사용할때는 children box에 적용하지 않는다. 오직 부모 클래스에만 flex를 적용한다. children box에 일일히 명령할 필요 없이 부모 클래스에게 한 번 명령하는 구조이다.

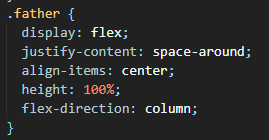
(2) flex는 기본적으로 inline-block의 형태를 갖는다.

(3)justify-content, align-itm을 이용하여 item의 위치를 조정할 수 있다.

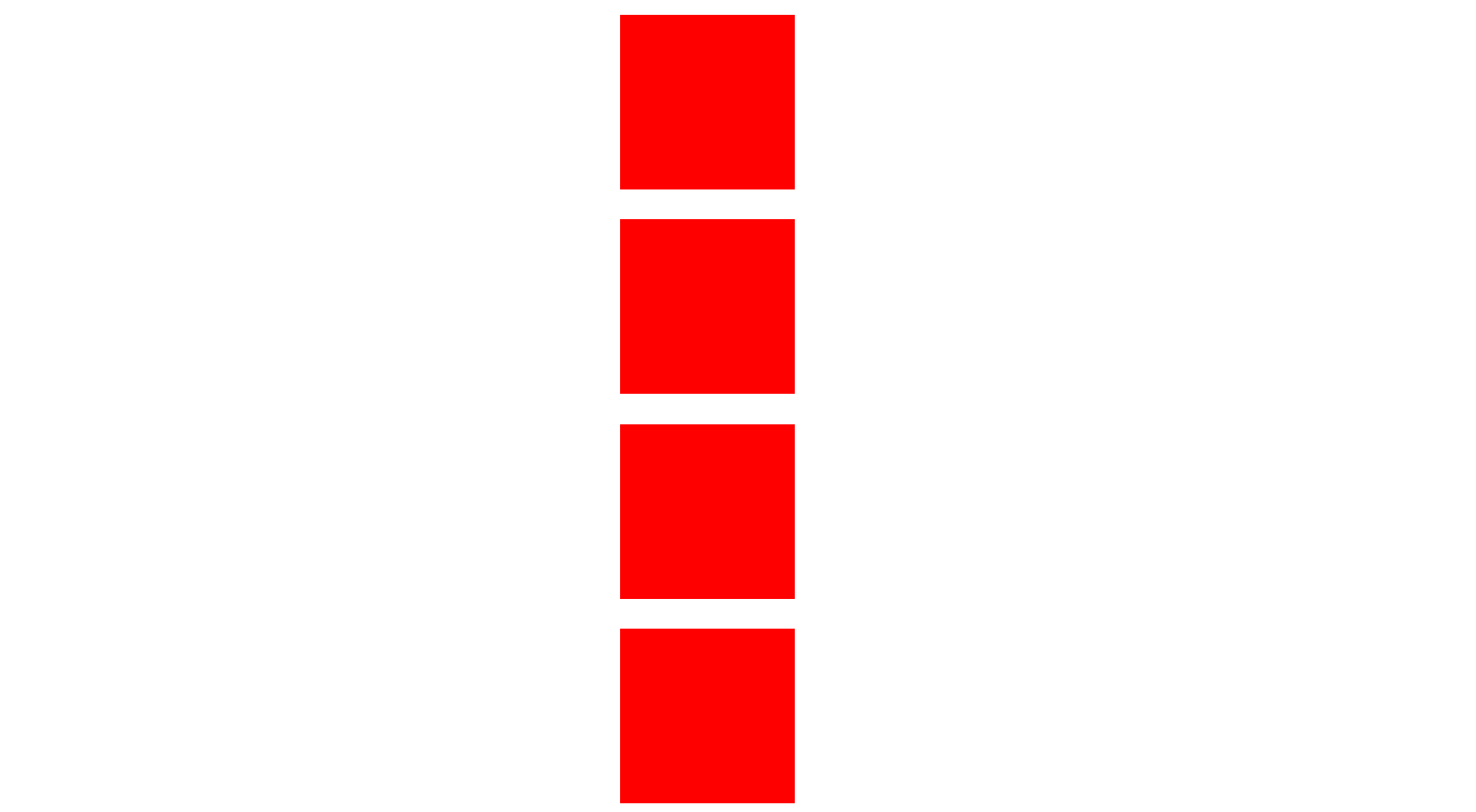




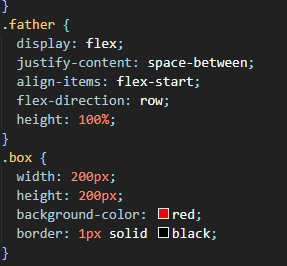
직관적으로 justify-content는 수평으로 align-item은 수직으로 적용이 되는 것을 확인할 수 있다.



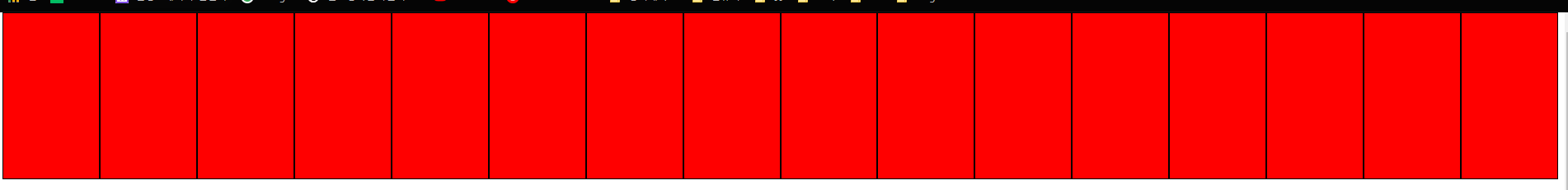
justify-content의 방향을 수직으로, align-item을 수평으로 하고 싶다면 하고 싶다면, 위와 같이 flex-direction을 column으로 설정해주면 된다.



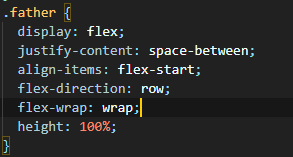
->Flex Container



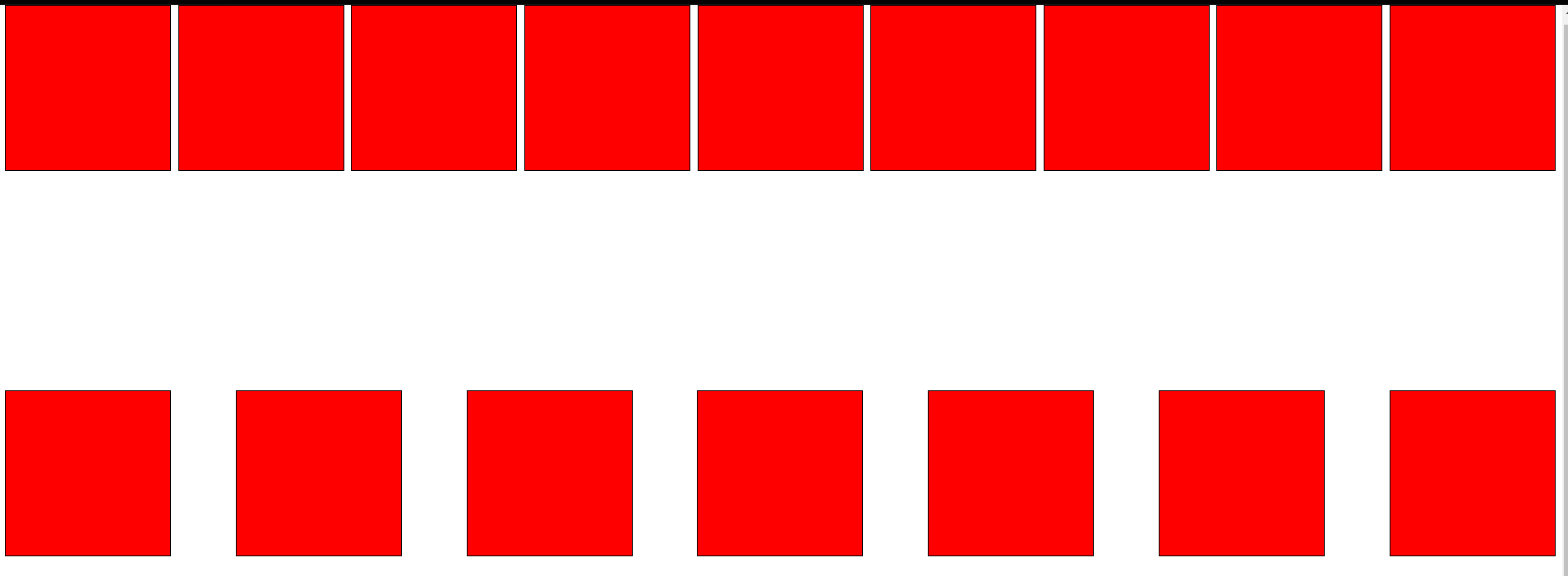
다음과 같이 flex container를 설정하고, children box에 border를 준 뒤 box를 여러 개 생성하면 다음과 같은 현상이 발생하는 것을 확인할 수 있다.



즉, Flex Container를 기준으로 창을 줄여도 그 창에 맞춰서 폭이 조정되는 것을 확인할 수 있다.

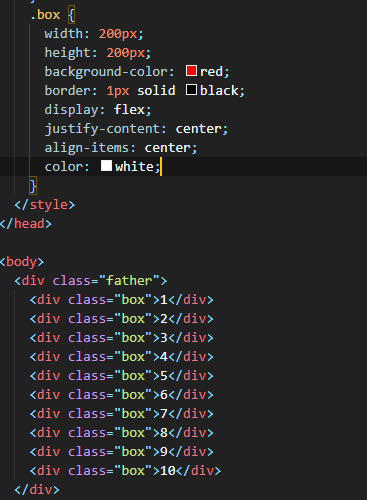


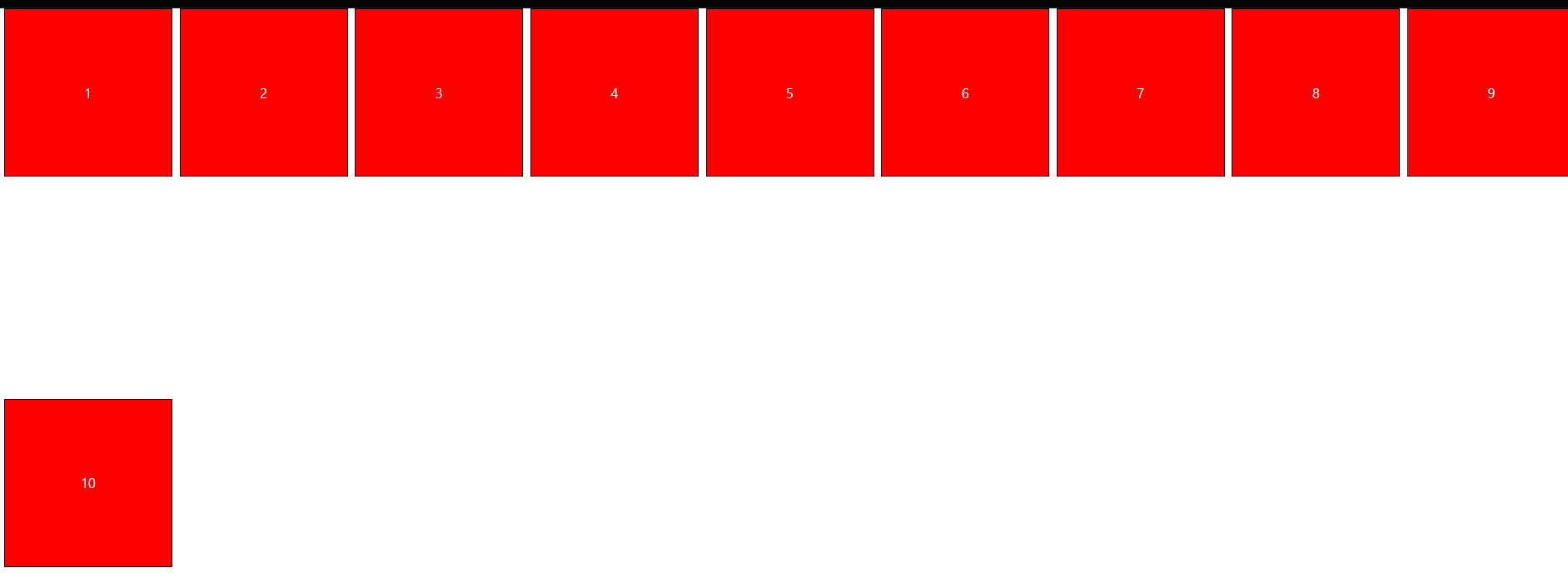
flex-wrap을 wrap으로 설정해주면, 설정한 값대로 box가 아래로 떨어지는 것을 확인할 수 있다. 즉, 디폴트 값은 no wrap으로 설정되어 있음을 알 수 있다.



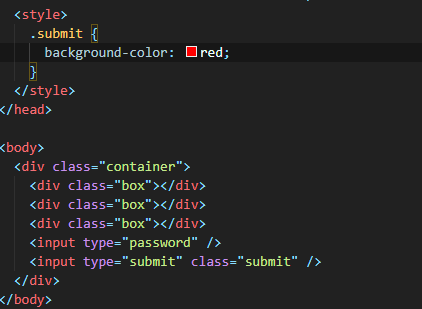
->children box flex

때로는 모든 박스들을 flex로 만들어, 해당 box안의 컨텐츠를 조정할수도 있다.

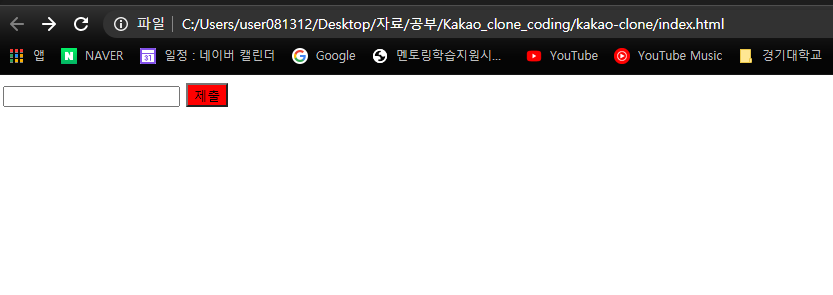




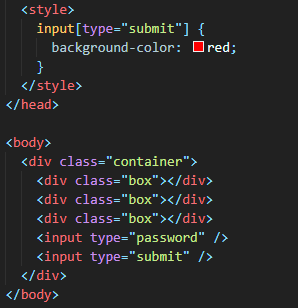
**\*CSS Selectors and Pseudo Selectors**



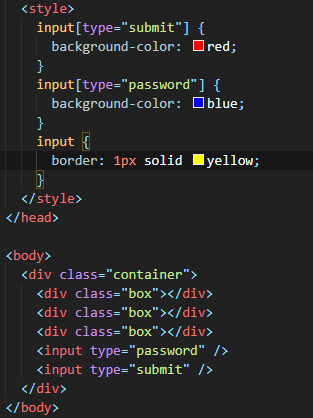
다음과 같이 비밀번호 입력 페이지를 만든다고 가정할 때, class를 submit으로 설정하고, 독립적인 style을 만들어 작성해 볼 수도 있다.



하지만 팀으로 일하는 경우 pseudo-selector(가상 셀렉터)를 통해서 선택하는 것이 좋다.

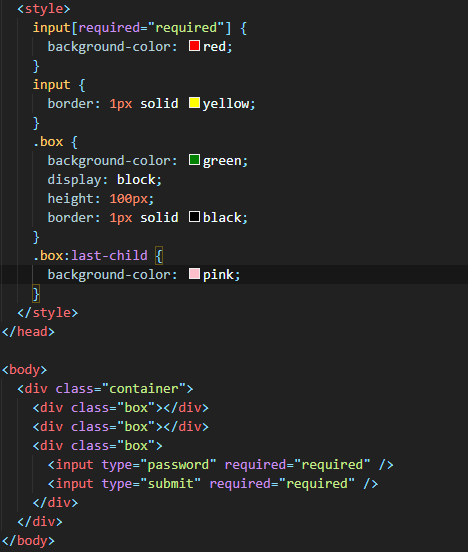


다음과 같이 input[속성=”값”]의 구조를 이용하여 CSS를 적용할 수 있다.

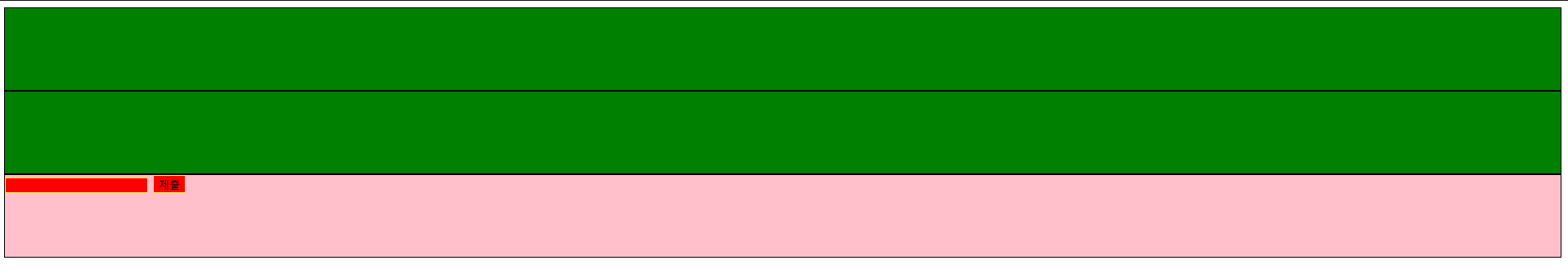


다음과 같이 활용해볼 수 있다.



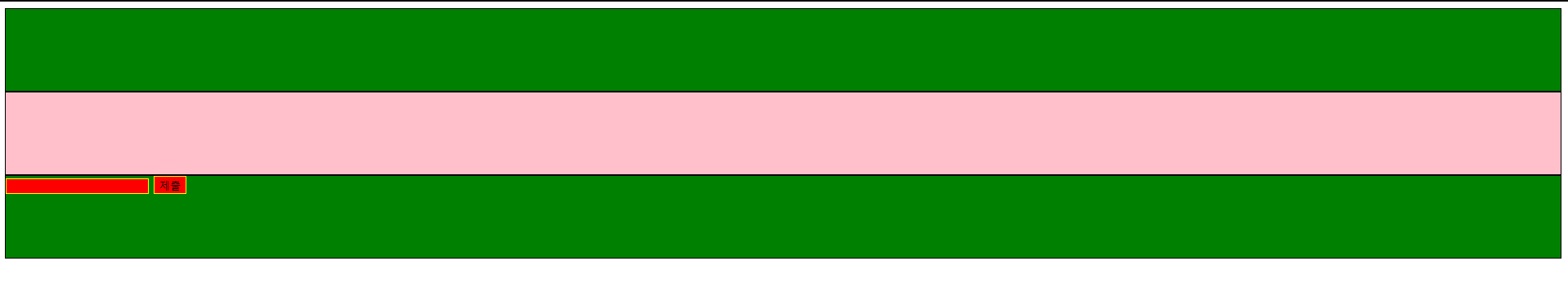


가상 셀렉터를 이용하면 다음과 같은 응용도 할 수 있다. 제일 마지막 박스 만을 last-child를 이용하여 선택하고, 배경색을 분홍색으로 지정한 예이다.



첫 번쨰는 first-child, n번쨰는 nth-child(번호)를 이용하여 작성할 수 있다.

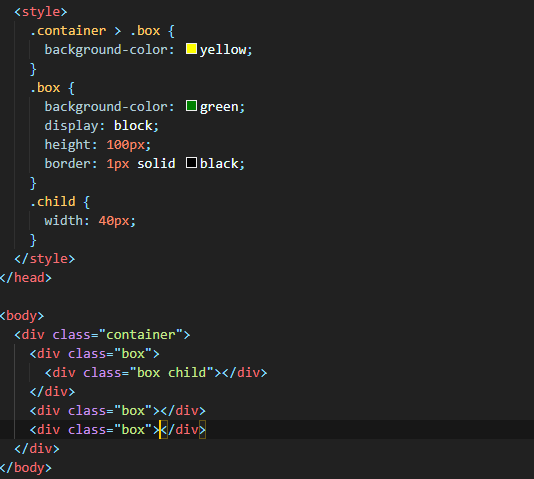




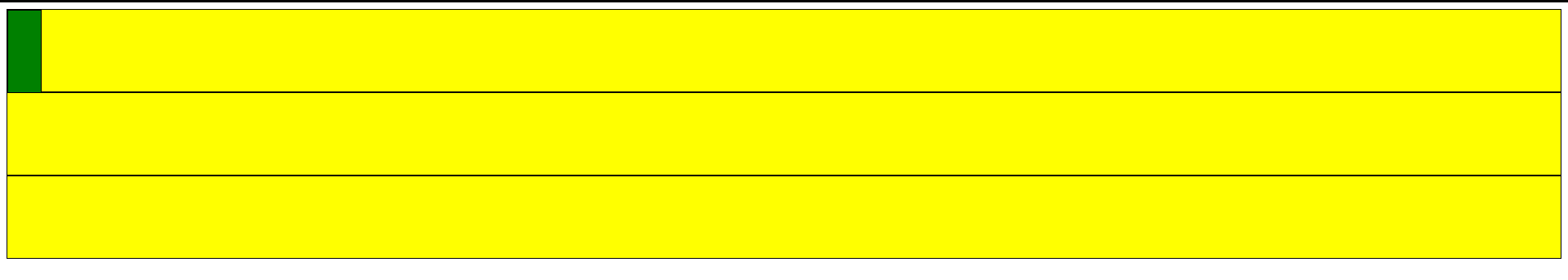
nth-child는 점화식을 이용하여 작성할수도 있다. nth-child(2n)이라고 작성하면 2번째 박스만을 선택한다.

태그 이름이나, Class, id를 쓰지않고 CSS를 적용하는 방법은 pseudo-selector(가상 셀렉터)가 존재한다는 것을 이해하자.

->가상 셀렉터 심화(직계 가상 셀렉터)

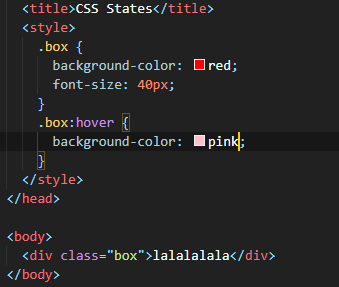


.container > .box와 같이 >를 이용하여 직계의 관계를 설정해 줄 수 있다. 위에서 box의 색은 초록색을 뜻하고 이에 따라 box와 box의 child box는 초록색이 되어야 함을 알 수 있다. 하지만 container의 직계 박스를 yellow라고 설정해 두었기 때문에, container의 직계 box는 노란색을 가져야한다.

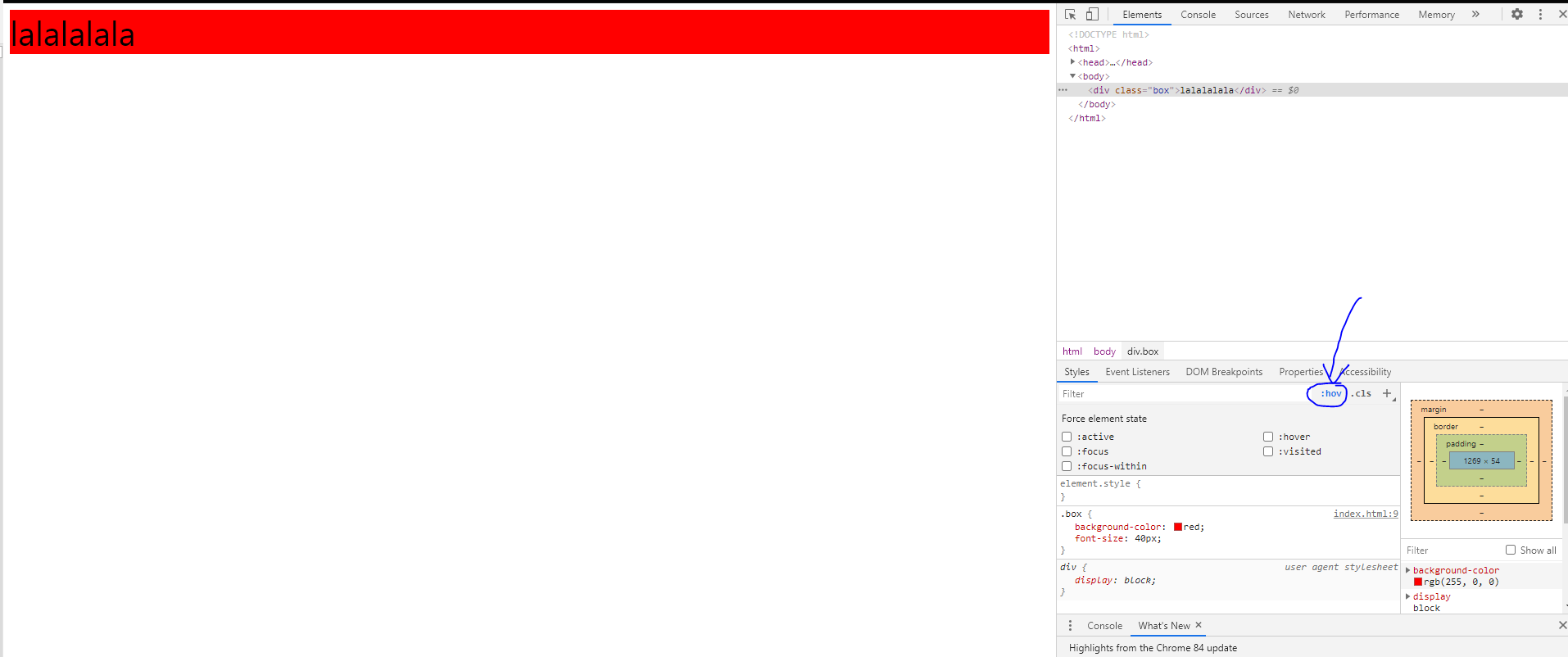


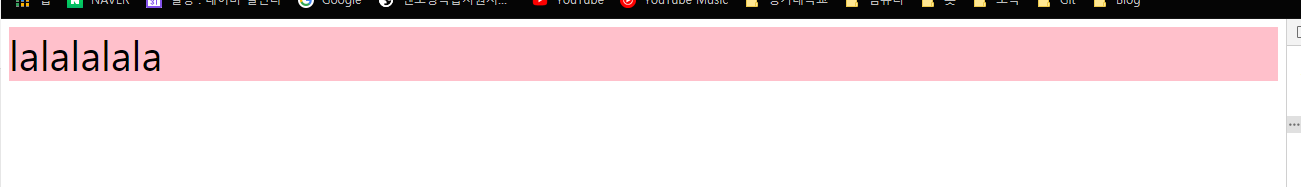
**\*Element States with CSS**

->hover state



해당 글씨 위에 마우스를 올려보면 배경색이 분홍색으로 변하는 것을 알 수 있다. 이것이 hover state를 의미하며, 요소 검사에서 hov탭을 클릭하여 확인할 수 있다.





->activate state : 클릭할 때 마다 상태가 변한다. hover state와 동시에 사용할수도 있다.



->foucus state : 클릭하면 모든 하위 명령(activate, hover…)을 무시하고 해당 설정으로 변화한다.



->visited state: 방문한 링크의 속성을 변경해준다.

