# CSS

html 만으로는 디자인이 별로...
- html + 새로운 디자인을 위한 TAG ⇒ 쉬운 길
- 디자인에 최적화된 새로운 언어 ⇒ 어려운 길 → CSS

- 먼저 쉬운길 (지금은 안써요) (디자인 TAG)
  <font color="color"> ~ </font>

(문제점)
1. 웹PAGE를 설명해주는 TAG와 속성에 디자인에 대한 정보가 비효율적..
2. 하나하나 바꿔야함..

등등 ⇒ CSS 활용하자!

---

주석 TAG (무시)

<!-- 내용 -->

지금까지 기본적으로 web browser는 html을 해석해서 처리

CSS는 html과 완전히 다른 컴퓨터 언어
=> html 문법으로 이것은 CSS니까 CSS로 해석해야 한다고 선언
=> style TAG 사용, <style> </style>

색바꾸기 (a TAG)

```
<style>
    a {
        color: red;
    }
</style>
```

세미콜론

⇒ 모든 A TAG에 해당

CSS 를통해서

- 가독성↑
- 중복 code 줄일 수 있음
- 유지보수가 용이함.
- 디자인에 있어 html보다 효율적.

=> <CSS부분> 디자인관련 / <html> Web의정보 ] → Web page 해석이 용이

(html은 정보에 전념)

CSS 기본문법

1. style TAG 사용

```
<style>
  a {
    color:black;
  }
</style>
```

→ 선택자(Selector) : 무엇을 꾸며줄까?

→ 효과 (declaration) : 선언

→ 세미콜론 : 효과구분자

CSS를 적용하는 두가지 방법

2. style 속성 사용

```
<ol>
  <li><a href="1.html">HTML</a></li>
  <li><a href="2.html" style="color:red">CSS</a></li>
</ol>
```

html의속성 : 그 값으로 CSS의 효과

=> Selector 사용할필요없다.

=> 세미콜론 구분자로 효과 여러개 구분가능.

<a>태그의 밑줄 없애기

text-decoration : none;

밑줄긋기

text-decoration : underline;

## 속성 스스로 알아내기 => 검색엔진을 이용, 외울필요가 없다! 뇌 혹사 X.

### 글자 키우기

font-size : 15px;
large;
150%;

### 글자 정렬

text-align : center;
left
right;
justify;

## 선택자 스스로 알아내기

방문 했던 곳 → 회색 / 안한곳 → 검정색 / 현재 방문하고 있는곳 → 하늘색

```
<style>
  a {
    color:black;
    text-decoration: none;
  }
  .saw {
    color:gray;
  }
  .active {
    color:red;
  }
  h1 {
    font-size:45px;
    text-align: center;
  }
</style>
</head>
<body>
<h1><a href="index.html">WEB</a></h1>
<ol>
  <li><a href="1.html" class="saw">HTML</a></li>
  <li><a href="2.html" class="saw active">CSS</a></li>
  <li><a href="3.html">JavaScript</a></li>
</ol>
```

순서대로.

.을통해 class가 saw인 태그를 가리키는 선택자가 됨.

class 속성을 통해 Grouping
class는 띄어쓰기로 구분, 여러개 가능

=> 그렇게 좋은 방법이 아니다. 같은 class에 있으면 상/하위 구분이 없어서 순서에 따라 결과 바뀜 => id = "active" 속성추가, active 선택자를 #active로 변경

우선순위 id > class > 그냥 TAG 선택자

## 왜 이러한 우선순위?

구체적인 것들부터 우선.

=> 포괄적인 선택자 (TAG선택자)를 통해 전체적인 속성을 작성한 후
구체적인 선택자(class, id)로 예외처리를 해낼 수 있음 => 훨씬 효율적.

알아보기 => css selector 검색

# 박스모델

※ 기본적으로 <h..> 태그는 줄바꿈
※ css 주석처리 /* ~ */
테두리 치기

```
h1{
    border-width : 5px
           -color : red
           -style : solid
}
```

=> 태그의 성격과 쓰임에 따라 차지하는 테두리가 다름을 알 수 있다.

=> 화면 전체를 차지하는 TAG → block level element
자기 자신의 content 크기 만큼 갖는 TAG → inline element

display : inline;
display : block;
display : none; → 사라짐
효과를 통해 상호 변환이 가능하다. (기본값일 뿐)

중복 제거..

(comma를 통해 여러 선택자를 선언 가능)

```
<style>
  h1, a{
    border:5px solid red;
  }
</style>
```

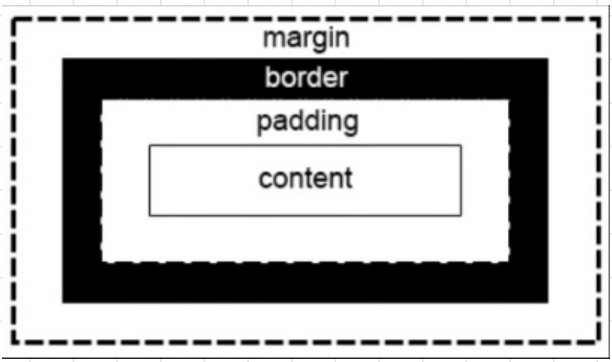
→ 효과들을 space로 구분하여 나열, 사용

padding을통해 content와 테두리 간격 조절 가능

padding:20px;

margine을 통해 테두리와 테두리 사이의 간격조절 가능

margine: 0;




width를 통해 TAG의 크기 변경 가능

width : 100px;

★중요한 기능 (CSS수정시 Good)

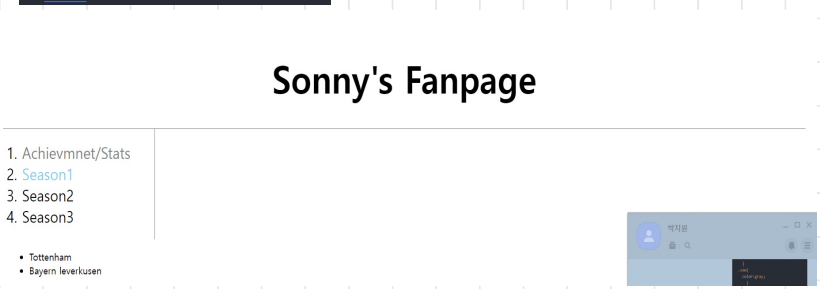
Web page 우클릭 → 검사

TAG선택시 어떠한 style 요소의 영향을 받고 있는지 알려줌.

=> CSS, html 이 복잡할 때 TAG가 어떠한 CSS의 영향을 받는지 효과적으로 볼 수 있다.

```
<style>
  body{
    margin: 40px;
     }
  a {
    color: black;
    text-decoration: none;
    }
  .saw{
    color:gray;
     }
  #active{
    color:skyblue;
        }
  h1 {
    font-size:60px;
    text-align:center;
    border-bottom:1px solid gray;
    margin:-7px;
    padding:40px;
     }
  ol {
    font-size: 25px;
    border-right:1px solid gray;
    width:220px;
    margin:7px;
    padding: 20px;
     }
</style>
```

=>나의 Web page에 적용



# 그리드

디자인 목적을 위해 의미를 갖지않고 레이아웃을 나눌때

★  ~  (block level)
 ~  (inline )

Style tag에서
display : grid;
grid-template-columns : 150px 1fr
1영역 , 2영역
화면 전체 사용

# 반응형 디자인

화면의 크기에 따라서 Web page의 각요소들이 반응해서 최적화된 모습으로 바뀌게 하는 것

미디어 쿼리 → 화면의 특성들에 따라서 어떠한 조건을 만족할때 CSS의 동작여부설정

```
<style>
  div{
    border:10px solid green;
    font-size:60px;
  }
  @media (max-width:800px) {
    div{
      display:none;
    }
  }
</style>
```

screen width < 800px 일때

div를 display:none으로 설정

# 그리드와 반응형 디자인 (미디어쿼리) 내코드에 적용

```
#grid {
 display:grid;
 grid-template-columns: 150px 1fr;
}
#grid ul {
 font-size: 20px;
 font-weight: bolder;
 border-right:1px solid gray;
 width:220px;
 margin:7px;
 padding: 20px;
 padding-left: 6px;
  }
#grid #information{
 padding-left: 120px;
}
@media(max-width:700px){
  #grid{
    display: block;
  }
  #grid ul{
    border-right:none;
  }
  #information{
    border-top:1px solid gray;
  }
```

grid를 이용해 같은 열에 배치

ul의 파라미터를 조정함으로써 보기 좋게 만듦

미디어 쿼리

- Web page 크기가 700px 이하일때
- id : grid 인 부분을 block level element로 변환
- id : grid의 ul 부분의 오른쪽 경계선 제거
- information 레이아웃의 윗쪽 경계선 설정

```
<div id = "grid">
  <ul>
    <li><a href ="Achievment.html">Achievmnet/Stats</a>
    <li><a href ="Season1.html">Season1</a>
    <li><a href ="Season2.html">Season2</a>
    <li><a href ="Season3.html">Season3</a>
    <li><a href = "Tottenham.html">Tottenham</a>
    <li><a href = "Leverkusen.html">Bayern leverkusen</a>
  </ul>
  <div id = "information">
    <h2>Son Heung Min</h2>
    <h3>Personal information<h3>
    <p><img src = "son.jpg" width = "50%"></p>
    <h3>Awards</h3>
    <p>2021-2022 Primer league Golden boot Winner</p>
    <h3>Recent Video</h3>
    <p><iframe width="560" height="315" src="https://www.you
  </div>
</div>
```

=> html 부분.

# CSS 코드의 재사용

Web page가 많을때 모두 동일한 CSS 코드를 붙여 넣는 일은 비효율적..
항상 중복되는 작업을 줄일것.. ⇒중복을 제거함으로써 가독성을 높이고 유지보수가 쉬워진다!

⇒ <link> 태그를 활용

1. style TAG 안의 CSS코드를 복사후 style.css 파일에 붙여넣기, 저장
2. 해당 page의 <style> TAG를 지우기.
3. 대신 <link rel="stylesheet" hrefs="style.css"> 삽입
   style.css를 다운로드해서 코드삽입.

검사 - Network 탭에서 확인 가능

Network 적 측면에서는 web page 안에 CSS code가 직접내장되어 있는것이 트래픽나, 효율적
→ caching 을 통해 해결할 수 있다.

Caching 이란? 저장하다 라는 뜻. 한번 style.css 라는 파일을 다운로드 받았다면
파일이 바뀌기 전까지는 컴퓨터에 저장해 놓았다가 요청시 가져와서
속도를 높일수 있다. (Network의 트래픽 적게 사용 가능)
⇒ 가급적 추천.

# 마무리 하며..

* CSS를 지탱하는 두 개의 뿌리 선택자와 속성 ⇒중요
  속성을 많이 알수록 더풍부한 표현력을 만들 수 있다.
  선택자를 많이 알수록 내가 알고있는 속성을 더 정확하게 표현할수있다.

공부한 것을 써먹을 타이밍
부족한 것들을 채워 나갈 것.

끝.