

비전공자를 위한 HTML/CSS

1. HTML 이해하기

1) HTML 소개

- HTML이란? HTML은 프로그래밍 언어로 웹 페이지를 만들 때 사용됩니다. 웹 사이트들이 모두 HTML을 사용해 만들어집니다.

- HTML의 의미와 특징 : HTML(Hyper Text Markup Language) 중 Hyper Text는 단순한 텍스트를 넘어서 웹 페이지의 특정 부분과 연결할 수 있는 기능을 가진 텍스트 즉, 링크를 의미, Markup Language는 프로그래밍 언어의 한 종류로 정보를 구조적, 계층적으로 표현 가능하다는 특징이 있습니다.

HTML은 파일 확장자로 .html을 쓰며, 그 파일 안에 html 코드를 작성하게 됩니다.

- HTML의 역사 : HTML은 1990년대 영국의 물리학자 팀 버너스 리가 제안하여 개발되었습니다. 초기 개발 목적은 연구소의 연구원들이 신속하게 정보와 문서를 공유하기 위해서였습니다. 현재의 웹 문서의 형태는 아니지만, 정보를 공유한다는 목적은 여전히 같습니다.

<https://developer.mozilla.org/ko/docs/Web/HTML>

2) HTML 문법 - 태그

- 태그란(Tag)? HTML은 태그들의 집합이며, 태그는 HTML에서 가장 중요하고 기본이 되는 규칙입니다. 태그는 '무언가를 표시하기 위한 꼬리표, 이름표'라는 의미가 있으며, HTML에서도 이와 비슷한 의미로 해석됩니다. 우리가 다양한 태그들을 이용해 코드를 작성하면, 브라우저가 이를 인식해 내용을 표현하게 됩니다.

- 태그를 사용하는 방법 : 태그는 <, > 기호로 표현하며 <, > 기호 사이에 태그 이름이 들어갑니다. 대부분 태그는 시작 태그와 종료 태그로 이루어지며 종료 태그는 태그 이름 앞에 '/' 기호가 붙습니다. 시작 태그와 종료 태그 사이에 실제 화면에 나타나는 내용이 위치하게 됩니다.

<h1>Hello, HTML</h1> -> 가장 기본적인 예로, <h1>을 사용해 'Hello, HTML'을 출력하는 코드

https://www.w3schools.com/tags/ref_byfunc.asp

- 요소란(Element)? 내용을 포함한 태그 전체를 요소(Element)라고 합니다. 태그와 요소는 의미가 다르지만 많은 사람이 태그와 요소를 같은 의미로 사용하니 혼동하지 않도록 주의해야 합니다.

<https://developer.mozilla.org/ko/docs/Web/HTML/Element>

3) HTML 문법 - 속성

- 속성이란(Attribute)? 속성은 태그에 추가로 정보를 제공하거나 태그의 동작이나 표현을 제어할 수 있는 설정값을 의미합니다.

- 속성을 사용하는 방법 : 속성은 이름과 값으로 이뤄져 있습니다. 시작 태그에서 태그 이름 뒤에 공백으로 구분 후 속성 이름 = "속성값"으로 표현합니다. 속성값은 홑따옴표(')와 쌍따옴표(")로 감싸 표현합니다.

<h1 id="title">Hello, HTML</h1> -> <h1>에 id 속성을 추가해 title 값을 선언한 코드

- 여러 속성을 사용하는 방법 : 의미와 용도에 따라 여러 속성이 존재하며 하나의 태그에 여러 속성을 선언할 수 있습니다. 여러 속성을 선언할 때는 공백으로 구분해서 사용합니다.

<h1 id="title" class="main">Hello, HTML</h1> -> <h1>에 id와 class 2개의 속성을 선언한 코드

속성의 선언 순서는 태그에 영향을 미치지 않으며 class를 id보다 먼저 선언해도 결과는 같습니다.

- 속성의 종류 : 속성은 종류에 따라 모든 태그에 사용할 수 있는 글로벌 속성과 특정 태그에서만 사용할 수 있는 속성으로 구분됩니다. 또한, 선택적으로 쓸 수 있는 속성과 특정 태그에서 필요한 필수 속성으로 구분됩니다. 예시에서 쓰인 id와 class 속성은 글로벌 속성입니다.

https://www.w3schools.com/tags/ref_standardattributes.asp

https://www.w3schools.com/tags/ref_attributes.asp

4) HTML 문법 - 태그 중첩

- 태그의 중첩(Nesting Tags) : 태그 안에 다른 태그를 선언할 수 있습니다. 태그를 중첩해서 사용 시 중첩되는 태그는 부모 태그를 벗어나서는 안 됩니다.

`<h1>Hello, <i>HTML</h1></i>` -> `<h1>`안에서 `<i>`가 시작되었으나, `<i>`보다 부모인 `<h1>`의 종료 태그가 먼저 선언, 엇갈리게 태그를 선언하는 것은 올바르지 않습니다.

`<h1>Hello, <i>HTML</i></h1>` -> `<i>`의 종료 태그를 먼저 선언해준 후 `<h1>`의 종료 태그를 선언해야 합니다.

<http://validator.kldp.org/>

5) HTML 문법 - 빈 태그

- 빈 태그란(Empty Tag)? 태그는 기본적으로 시작 태그와 종료 태그 2개가 1쌍으로 이뤄져 있으며, 그 사이에 내용이 들어가게 됩니다. 하지만, 태그 중에는 그렇지 않은 태그가 존재하기도 하는데 그런 태그를 내용이 없는 빈 태그라고 합니다.

`
`, ``, `<input type="">` 빈 태그는 내용이 없어서 종료 태그가 필요하지 않습니다.

- 빈 태그의 특징 : 빈 태그는 내용만 비어있을 뿐 속성을 통해서 화면에 나타내거나 화면에 표시되지 않더라도 다른 용도로 사용되는 태그입니다. 빈 태그의 대표적인 경우는 브라우저가 직접 화면에 내용을 그려줘야 하는 경우입니다. 이런 태그는 브라우저가 내용을 대체한다고 하여 replacement 태그, 대체되는 태그라고 합니다. 빈 태그에 대체되는 태그만 있는 것은 아니며 실제로 화면에 출력될 내용이 없어 다른 용도로 쓰이는 태그도 존재합니다. 예시 코드의 `
`이 바로 이 경우입니다.

https://developer.mozilla.org/en-US/docs/Glossary/Empty_element

6) HTML 문법 - 공백

- HTML에서의 공백(Space) : 기본적으로 HTML은 두 칸 이상의 공백을 모두 무시합니다.

`<h1>Hello, HTML</h1>`

`<h1>Hello, HTML</h1>`

`<h1>`

 Hello,

 HTML

`</h1>`

-> HTML은 두 칸 이상의 공백과 개행을 모두 무시하기 때문에 위 세가지 모두 같은 텍스트가 화면에 나타나게 됩니다.

강좌에서 사용하는 에디터 : <https://atom.io/>

<https://www.hongkiat.com/blog/change-default-text-wrapping-html-css/>

7) HTML 문법 - 주석

- HTML에서의 주석(Comment Tags) : 주석은 화면에 노출되지 않고 메모의 목적으로만 사용하는 것을 의미합니다. HTML 파일 내에 주석으로 표시해주면 브라우저는 해당 부분을 인식해 해석하지 않습니다. 주석의 시작은 `<!--`로 표시하고, `-->` 표시로 종료합니다.

`<!-- 여기에 작성되는 내용들은 모두 주석 처리가 됩니다. -->`

`<!--` 주석은 여러 줄로 작성할 수도 있습니다.

`<h1>Hello, HTML</h1>`

 위 `<h1>`태그는 브라우저가 해석하지 않습니다.

`-->`

https://www.w3schools.com/tags/tag_comment.asp

8) 문서의 기본 구조

- HTML의 기본 구조(HTML Document) : HTML의 기본 구조는 웹 문서를 작성할 때 반드시 들어가야 하는 기본적인 내용으로 크게는 문서 타입 정의와 <html>요소로 구분합니다.

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
  <head>
```

```
    <meta charset="UTF-8">
```

```
    <title>HTML</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1>Hello, HTML</h1>
```

```
  </body>
```

```
</html>
```

- 문서 타입 정의 : 문서 타입 정의는 보통 DTD(doctype)라고 부릅니다. 이 문서가 어떤 버전으로 작성되었는지 브라우저에 알려주는 선언문이며 반드시 문서 내 최상단에 선언되어야 합니다.

- <html> 요소 : 문서 타입 선언 후에는 <html>태그가 나와야 하고, 자식으로는 <head>태그와 <body>태그가 있습니다. <html> 태그의 lang 속성은 문서가 어느 언어로 작성되었는지를 의미합니다. <head>태그에 위치하는 태그들은 브라우저 화면에 표시되지 않습니다. 대신 문서의 기본 정보 설정이나 외부 스타일 시트 파일 및 js 파일을 연결하는 등의 역할을 합니다. <meta> 태그의 charset 속성은 문자의 인코딩 방식을 지정합니다. <body>태그에는 실제 브라우저 화면에 나타나는 내용이 들어가며, 앞으로 우리가 다루는 태그들 대부분이 모두 여기에 해당이 됩니다. 위 코드는 가장 기본적인 문서 구조로, 보통은 이보다 더 많은 태그가 들어가게 됩니다.

https://www.w3schools.com/tags/tag_doctype.asp

https://www.w3schools.com/tags/tag_html.asp

https://www.w3schools.com/tags/tag_head.asp

https://www.w3schools.com/tags/tag_body.asp

https://www.w3schools.com/tags/tag_meta.asp

2. HTML 태그

1) HTML 태그 소개

- HTML 버전이 업그레이드 되면서 태그가 새로 추가되기도 하고 삭제되기도 합니다. 현재 태그의 개수는 대략 130여 개 정도입니다. 관련 통계 사이트를 보면, 실제로 대다수의 웹 페이지는 대략 25개 정도의 다른 태그가 사용된다고 합니다. <https://www.advancedwebranking.com/html/#overview>

- html, head, body, title 등 기본적으로 들어가는 태그까지 포함되기 때문에, 실제로 주로 사용하는 태그는 이보다 더 적을 것으로 예상합니다.

<http://html5doctor.com/element-index/>

<https://www.w3schools.com/tags/default.asp>

2) 제목과 단락요소

- 제목 태그(Heading) : 문서 내에 제목을 표현할 때 사용하는 태그입니다. 태그 이름은 heading을 줄여서 h로 쓰며, 제목의 레벨에 따라서 <h1>~<h6>까지 있습니다. 보통 <h1>은 해당 페이지를 대표하는 큰 제목으로 주로 사용되며, 숫자가 올라갈수록 조금 더 낮은 수준의 소제목이 나타내게 됩니다. 하지만, 현재 웹 페이지의 내용은 제목과 본문 식의 문서 형태보다는 주로 이미지나 그림처럼 시각적인 형태로 표현되고 있어서 제목 태그를 <h6>까지 쓰는 경우는 거의 없습니다.

<h1>역사</h1>

<h2>개발</h2>

1980년, 유럽 입자 물리 연구소(CERN)의 계약자였었던 물리학자 팀 버너스리가 HTML의 원형인 인콰이어를 제안하였다.

... 이하 생략

<h2>최초 규격</h2>

HTML 최초의 일반 공개 설명은 1991년 말에 버너스리가 처음으로 인터넷에서 문서를 "HTML 태그"(HTML tag)로 부르면서 시작되었다.

... 이하 생략

-> 예시 글에서는 역사가 가장 큰 제목이기 때문에 <h1>를 사용했고, 개발과 최초 규격에는 그보다 아래 단계인 <h2>를 사용했습니다. 원문을 참고하여 적절하게 태그를 사용했습니다. (<https://ko.wikipedia.org/wiki/HTML> 중 역사 섹션) 제목 태그를 사용하면, 일반 텍스트보다 더 강조되는 스타일이 적용되는데 이는 브라우저가 기본적으로 제목 태그에 제공하는 스타일입니다. 제목이라는 의미에 맞게 기본적으로 좀 더 굵고 크게 표현이 됩니다.

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements

<https://www.w3.org/MarkUp/html3/headings.html>

- 단락 태그(paragraph)는 paragraph를 줄여서 p로 씁니다. 제목 태그에 이어 본문에 해당하는 내용을 단락 태그를 이용해 작성해보겠습니다.

<h1>역사</h1>

<h2>개발</h2>

<p>

1980년, 유럽 입자 물리 연구소(CERN)의 계약자였었던 물리학자 팀 버너스리가 HTML의 원형인 인콰이어를 제안하였다.

... 이하 생략

</p>

<h2>최초 규격</h2>

<p>

HTML 최초의 일반 공개 설명은 1991년 말에 버너스리가 처음으로 인터넷에서 문서를 "HTML 태그

"(HTML tag)로 부르면서 시작되었다.

... 이하 생략

</p>

-> 화면에는 별다른 변화는 없지만, 이전보다 훨씬 의미에 맞게 잘 짜인 마크업 구조라고 볼 수 있습니다.

- 개행 : <p>를 사용해서 단락으로 구분하면 자연스럽게 개행이 됩니다. HTML 문법 중 공백 시간에 배운 대로, html은 한 칸 이상의 공백 및 개행을 무시하기 때문에 실제 코드창에 개행을 하더라도 화면에 나타나지는 않습니다. 따라서 개행을 위해 쓰이는 태그가 바로
입니다. (linebreak 를 줄여서 br 이라고 합니다.)
은 닫기 태그와 내용이 존재하지 않는 빈 태그입니다. 개행하고자 하는 곳에서
을 선언하면 개행이 됩니다. 이처럼 <p>를 이용하면 태그 자체에서 자연스럽게 개행이 되지만, <p> 내부에서 강제로 개행을 하기 위해서는
을 이용해야 합니다.

3) 텍스트를 꾸며주는 요소

- 텍스트 표현 태그 : 웹 표준화가 대두하면서 웹 문서의 구조와 표현을 분리하였습니다. 그 과정에서 많은 표현용 태그들이 사라졌고, 지금은 표현용 태그가 얼마 남지 않았습니다.

 : bold 태그는 글자를 굵게 표현하는 태그입니다.

<i> : italic 태그는 글자를 기울여서 표현하는 태그입니다.

<u> : underline 태그는 글자의 밑줄을 표현하는 태그입니다.

<s> : strike 태그는 글자의 중간선을 표현하는 태그입니다. (예전에 존재했던 strike 태그와는 다른 태그로, strike 태그는 폐기되어 더는 사용할 수 없습니다.)

-> 위 태그들은 의미가 없는 표현용 태그이기 때문에 사용하실 때는 각별히 주의를 하셔야 합니다.

살펴본 태그 이외에도 텍스트 관련 태그는 많습니다.

<p>

Lorem <i>ipsum</i> dolor sit amet

<u>Lorem</u> <s>ipsum</s> dolor sit amet

</p>

https://developer.mozilla.org/en-US/docs/Web/HTML/Element#inline_text_semantics

4) 앵커 요소

- <a> : <a>(anchor 태그)는 a태그, 앵커, 링크 등 여러 이름으로 불립니다.

네이버

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a>

- href 속성 : 링크를 만들기 위해 <a>는 반드시 href(hypertext reference) 속성을 가지고 있어야 합니다. href 속성의 값은 링크의 목적지가 되는 URL입니다.

- target 속성 : target 속성은 링크된 리소스를 어디에 표시할지를 나타내는 속성입니다. 속성값으로는 _self, _blank, _parent, _top이 있습니다. _self는 현재 화면에 표시한다는 의미로, target 속성이 선언되지 않으면 기본적으로 self와 같이 동작합니다. _blank는 새로운 창에 표시한다는 의미로 외부 페이지가 나타나게끔 하는 속성입니다. _parent와 _top은 프레임이라는 특정 조건에서만 동작하는 속성으로, 최근에는 프레임을 잘 쓰지 않기 때문에 따로 다루지 않고 넘어가겠습니다.

- 기타 속성 : <a>에는 이 외에도 많은 속성이 있습니다. 아래 링크를 참고해 다루지 않은 나머지 속성을 직접 공부하기를 권장합니다.

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a>

(참고로 속성 이름 옆 html5 레이블로 표시된 것은 html5 버전이 나오면서 새로 생긴 속성이라는 뜻입니다. Obsolete attributes로 표시된 부분은 폐기된 속성들이므로 보지 않으셔도 됩니다.)

- 내부링크 : <a>를 통해 만들어진 링크가 꼭 외부 페이지로만 이동하는 것은 아닙니다. <a>를 통해 페이지 내부의 특정 요소로 초점을 이동할 수도 있는데, 이를 내부링크라고 합니다. 내부링크를 사용할 때는 href 속성값에 #을 쓰고 그 뒤에 페이지 내에서 이동하고자 하는 요소의 id 속성값을 적으면 됩니다.

```
<a href="#some-element-id">회사 소개로 이동하기</a>
```

... 중략.

```
<h1 id="some-element-id">회사 소개</h1>
```

보통 페이지에 내용이 많아 스크롤이 길어질 경우, 빠르게 화면 최상단으로 이동하고자 할 때 내부 링크를 주로 사용합니다. 웹페이지에서 화면 하단에 있는 'top' 또는 '맨 위로 이동하기' 버튼이 이에 해당합니다.

5) 의미가 없는 컨테이너 요소

- 의미가 없는 컨테이너 요소 : 태그 자체에 아무 의미가 없으며, 단순히 요소들을 묶기 위해 사용되는 태그입니다. 스타일을 주거나 서버에 보내는 데이터를 담기 위한 용도로 이런 의미 없는 요소들이 사용됩니다. 이런 의미 없는 태그의 사용빈도는 매우 높습니다. 그 이유는 html 태그들은 문서를 웹에 나타내기 위해 제작되어 기본적으로 문서에 최적화되어있는의미를 지니는데, 현재 웹의 형태는 문서 형태에서 많이 벗어났기 때문입니다. 다행히 HTML 버전이 업그레이드되면서 웹에 알맞은 태그들이 많이 생겼습니다. 가장 대표적으로 많이 쓰이는 의미가 없는 태그는 <div>, 입니다.

- <div>태그와 태그 : div(division) 태그는 블록 레벨 태그입니다. 블록 레벨 요소는 기본적으로 한 줄을 생성해서 내용을 표현합니다. 반면, span 태그는 인라인 레벨 태그입니다. 인라인 레벨 요소들은 블록 레벨 요소의 한 줄 안에서 표현되는 요소들입니다. 이전에 배운 <p>는 블록 레벨 태그이며, 그 안에서 텍스트를 꾸며주는 , <i>, <u>, <s>는 모두 인라인 레벨 태그입니다.

```
<div>
```

```
    <span>Lorem</span> ipsum dolor sit.
```

```
</div>
```

<div>, 는 모두 아무 의미가 없으므로 실제 브라우저도 별다른 스타일을 적용하지 않습니다.

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div>

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/span>

6) 리스트 요소

- : ul(unordered list) 태그는 순서가 없는 리스트를 표현할 때 사용합니다.

```
<ul>
```

```
    <li>콩나물</li>
```

```
    <li>파</li>
```

```
    <li>국 간장</li>
```

```
    ...
```

```
</ul>
```

-> 콩나물국에 들어가는 일부 재료들을 나열한 리스트입니다. 각 재료는 나오는 순서가 바뀌어도 상관없으므로 이러한 것은 순서가 없는 리스트로 표현할 수 있습니다. 을 선언한 후 그 안에서 를 사용해 각 항목을 나타내서 사용합니다.

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ul>

- : ol(ordered list) 태그는 순서가 있는 리스트를 표현할 때 사용합니다.

```
<ol>
```

```
<li>냄비에 국물용 멸치를 넣고 한소끔 끓여 멸치 육수를 7컵(1,400ml) 만든다.</li>
<li>콩나물을 넣고 뚜껑을 덮어 콩나물이 익을 때까지 끓인다.</li>
<li>뚜껑을 열고 대파, 마늘, 고춧가루를 넣고 끓인다.</li>
```

...

```
</ol>
```

-> 콩나물국을 끓이는 순서를 나열한 리스트입니다. 이 순서는 서로 바뀌면 안 되기 때문에 순서가 있는 을 사용해야 합니다. 을 선언한 후 그 안에서 를 사용해 각 항목을 나타내서 사용합니다.

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/ol>

- <dl> : dl(definition/description list) 태그는 용어와 그에 대한 정의를 표현할 때 사용합니다. <dl>은 앞서 배운 , 과는 구조가 조금 다릅니다. , 은 항목을 단순히 나열하는 구조지만, <dl>은 용어와 설명이 하나의 세트로 항목을 이루고 하나 이상의 항목으로 리스트가 이루어지는 구조입니다.

```
<dl>
```

```
<dt>리플리 증후군</dt>
```

```
<dd>허구의 세계를 진실이라 믿고 거짓된 말과 행동을 상습적으로 반복하는 반사회적 성격장애를 뜻하는 용어</dd>
```

```
<dt>피그말리온 효과</dt>
```

```
<dd>타인의 기대나 관심으로 인하여 능률이 오르거나 결과가 좋아지는 현상</dd>
```

```
<dt>언더독 효과</dt>
```

```
<dd>사람들이 약자라고 믿는 주체를 응원하게 되는 현상</dd>
```

```
</dl>
```

```
<dt> : 용어를 나타내는 태그
```

```
<dd> : 용어에 대한 정의 또는 설명을 나타내는 태그
```

용어 하나에 여러 정의가 들어갈 때, <dd>를 한 개 이상 쓰는 것이 가능합니다.

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dl>

7) 이미지 요소

- : 는 문서에 이미지를 삽입하는 태그로, 닫는 태그가 없는 빈 태그입니다.

```

```

- src 속성 : 의 필수 속성으로 이미지의 경로를 나타내는 속성입니다.

- alt 속성 : 이미지의 대체 텍스트를 나타내는 속성입니다. 대체 텍스트는 이미지를 대체하는 글을 뜻하며, 웹 접근성 측면에서 중요한 속성입니다. src 속성과 더불어 반드시 들어가야 하는 속성입니다.

- width/height 속성 : 이미지의 가로/세로 크기를 나타내는 속성입니다. 값의 단위는 필요하지 않으며, 값을 입력하면 자동으로 픽셀 단위로 계산됩니다. width/height는 선택적인 속성이지만 이미지의 크기가 고정적이라면 width/height 속성을 선언하는 것이 성능적인 측면에서 좋습니다. width/height 속성이 없으면 이미지는 원본 크기대로 노출되며, 둘 중 하나만 선언하면 나머지 한 속성은 선언한 속성의 크기에 맞춰 자동으로 비율에 맞게 변경됩니다. 반면 두 속성 모두 선언하면 이미지는 비율과 무관하게 선언한 크기대로 변경됩니다.

- 상대경로와 절대경로 : src 속성에는 이미지의 경로가 들어가며, 이미지의 상대경로와 절대경로가 있습니다. 상대경로는 현재 웹 페이지를 기준으로 상대적으로 이미지의 위치를 나타내는 경로이고, 절대경로는 실제 그 이미지가 위치한 곳의 전체 경로입니다.

```
<!-- 상대경로 -->
```

```

```

```
<!-- 절대경로 -->
```


-> 상대 경로에서의 './'는 페이지가 있는 현재 폴더를 나타냅니다.

- 이미지 파일 형식

gif : 제한적인 색을 사용하고 용량이 적으며 투명 이미지와 애니메이션 이미지를 지원하는 형식

jpg : 사진이나 일반적인 그림에 쓰이며 높은 압축률과 자연스러운 색상 표현을 지원하는 형식(투명을 지원하지 않는다.)

png : 이미지 손실이 적으며 투명과 반투명을 모두 지원하는 형식

8) 테이블 요소 1

- 표의 구성 요소 : 표는 셀(내용이 들어가는 하나의 칸)로 이루어져 있습니다. 표의 행(가로 방향)을 row, 열(세로 방향)을 column이라 합니다.

<table> : 표를 나타내는 태그

<tr> : 행을 나타내는 태그

<th> : 제목 셀을 나타내는 태그

<td> : 셀을 나타내는 태그

하나의 <table>은 하나 이상의 <tr>로 이루어져 있으며, <tr>은 셀을 나타내는 <th>, <td>를 자식으로 가지게 됩니다. 표를 구성할 때는 위에서 밑으로, 좌측에서 우측으로 작성하면 됩니다.

<table>

<tr>

<td>1</td>

<td>2</td>

<td>3</td>

<td>4</td>

</tr>

<tr>

<td>5</td>

<td>6</td>

<td>7</td>

<td>8</td>

</tr>

<tr>

<td>9</td>

<td>10</td>

<td>11</td>

<td>12</td>

</tr>

<tr>

<td>13</td>

<td>14</td>

<td>15</td>

<td>16</td>

</tr>

</table>

-> 위는 1부터 16을 표현한 4행 4열의 표를 나타내는 코드입니다. 브라우저 화면을 확인해보면 테두리가 없어 표가 어색해 보일 수 있습니다. 브라우저에서 제공하는 테이블의 기본 스타일에는 테두리가 없기 때문입니다. 확인을 위해 아래의 CSS 코드를 <head> 안에 입력하면 테두리가 나타나는 걸 확인할 수 있습니다.

```
<style>
```

```
th, td { border: 1px solid; }
```

```
</style>
```

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/table>

- 표의 구조와 관련된 태그 : 표가 복잡해지면 표를 해석해서 음성으로 전달해야 하는 스크린 리더기와 같은 보조 기기를 통해서는 표의 내용을 이해하는 것이 더욱 어려워질 것입니다. 따라서 표를 구조적으로 파악하기 위해 도움이 되는 태그를 사용해야 합니다.

<caption>: 표의 제목을 나타내는 태그

<thead>: 제목 행을 그룹화하는 태그

<tfoot>: 바닥 행을 그룹화하는 태그

<tbody>: 본문 행을 그룹화하는 태그

```
<table>
```

```
  <caption>Monthly Savings</caption>
```

```
  <thead>
```

```
    <tr>
```

```
      <th>Month</th>
```

```
      <th>Savings</th>
```

```
    </tr>
```

```
  </thead>
```

```
  <tbody>
```

```
    <tr>
```

```
      <td>January</td>
```

```
      <td>$100</td>
```

```
    </tr>
```

```
    <tr>
```

```
      <td>February</td>
```

```
      <td>$80</td>
```

```
    </tr>
```

```
  </tbody>
```

```
  <tfoot>
```

```
    <tr>
```

```
      <td>Sum</td>
```

```
      <td>$180</td>
```

```
    </tr>
```

```
  </tfoot>
```

```
</table>
```

-> <tfoot>은 <thead> 다음에 있지만, 실제 화면에서는 표의 맨 아래에 위치하게 된다는 점을 주의해야 합니다.

9) 테이블 요소 2

- 복잡한 표 만들기

```
<table>
  <caption>Specification values</caption>
  <thead>
    <tr>
      <th rowspan="2">Grade.</th>
      <th rowspan="2">Point.</th>
      <th colspan="2">Strength.</th>
      <th rowspan="2">Percent.</th>
    </tr>
    <tr>
      <th>kg/mm</th>
      <th>lb/in</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Hard</td>
      <td>0.45</td>
      <td>56.2</td>
      <td>80,000</td>
      <td>20</td>
    </tr>
    <tr>
      <td>Medium</td>
      <td>0.45</td>
      <td>49.2</td>
      <td>70,000</td>
      <td>25</td>
    </tr>
    <tr>
      <td>Soft</td>
      <td>0.45</td>
      <td>42.2</td>
      <td>60,000</td>
      <td>30</td>
    </tr>
  </tbody>
</table>
```

- <colgroup>, <col>, scope 속성, header 속성

10) 폼 요소 1

- 폼 요소는 서버에 데이터를 전달하기 위한 요소이며 <input>은 대표적인 폼 요소입니다. <input>은 내용이 없는 빈 요소이며 type 속성을 통해 여러 종류의 입력 양식으로 나타나게 됩니다. 본 강의에서는 자

주 사용되는 type들만 다루니 나머지는 아래 링크를 참고하시기 바랍니다.

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input>

- type="text"

<input type="text" placeholder="o o o">

-> 주로 아이디, 이름, 주소, 전화번호 등 단순한 텍스트를 입력할 때 사용합니다. type="text"에는 placeholder 속성이 존재합니다. placeholder 속성은 사용자가 입력하기 전 미리 화면에 노출하는 값으로, 입력하는 값의 양식을 표현할 수 있습니다.

- type="password"

<input type="password">

-> 암호와 같이 공개할 수 없는 내용을 입력할 때 사용합니다. 화면에는 type="text"와 같게 나타나지만 실제로 입력할 때 값을 노출하지 않습니다.

- type="radio"

<input type="radio" name="gender"> 남자

<input type="radio" name="gender"> 여자

-> 라디오 버튼을 만들 때 사용합니다. 라디오 버튼은 중복 선택이 불가능하며, 하나의 항목만을 선택해야 합니다.

- type="checkbox"

<input type="checkbox" name="hobby"> 등산

<input type="checkbox" name="hobby"> 독서

<input type="checkbox" name="hobby"> 운동

-> 체크박스를 만들 때 사용합니다. 체크박스는 중복 선택이 가능합니다.

- 라디오 버튼과 체크박스에는 checked, name 속성이 존재합니다. checked 속성은 값이 별도로 존재하지 않는 boolean 속성입니다. boolean 속성은 true/false 둘 중 하나의 값을 가지며 속성이 존재하면 true, 속성이 존재하지 않으면 false를 가집니다. name 속성은 라디오 버튼과 체크박스를 그룹화시켜주는 속성입니다. 아직 name 속성을 다루기 전이므로 실제 버튼을 클릭했을 경우 동작하는 방식에 대해서만 알아두시기 바랍니다.

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element#forms>

11) 폼 요소 2

- type="file"

<input type="file">

-> 파일을 서버에 올릴 때 사용합니다. 브라우저에 따라 표현되는 형태는 조금씩 다르지만, 모두 같은 역할을 합니다.

- type="submit|reset|image|button"

<form action="/test.html">

메시지: <input type="text" name="message">

<input type="submit">

<input type="reset">

<input type="image" src="http://placeholder.it/50x50?text=click" alt="click" width="50" height="50">

<input type="button" value="버튼">

</form>

-> submit, reset, image, button 타입은 모두 클릭 가능한 버튼을 만듭니다.

submit : form의 값을 전송하는 버튼

reset : form의 값을 초기화하는 버튼
 image : 이미지를 삽입할 수 있는 버튼 (submit과 동작이 동일함)
 button : 아무 기능이 없는 버튼
 이미지 버튼은 이미지 관련 속성인 src, alt 속성이 반드시 필요하며 width/height 속성을 적용할 수도 있습니다.

12) 폼 요소 3

- <select> : <select>는 선택 목록 상자 또는 콤보박스라고도 합니다. 몇 개의 선택지를 리스트 형태로 노출하고 그중 하나를 선택할 수 있게 하는 태그입니다. (multiple 속성을 사용하면 다중 선택도 가능합니다.)

```
<select>
  <option>서울</option>
  <option>경기</option>
  <option>강원</option>
  ...
```

```
</select>
```

-> <select>내부의 <option>으로 각 항목을 나타냅니다. <option>의 속성으로는 selected가 있으며 이는 선택된 항목을 의미합니다.

- <textarea> : 한 줄만을 입력할 수 있는 <input type="text">와 달리 여러 줄의 텍스트를 입력할 때 사용합니다.

```
<textarea rows="5" cols="30">
```

```
...
```

```
</textarea>
```

<textarea>에는 텍스트 상자의 크기를 조절하는 rows, cols 속성이 있습니다.

cols : 가로 크기를 조절하는 속성(한 줄에 들어가는 글자의 수, 수치의 의미는 평균적인 글자의 너비로 정확히 글자 수를 나타내지는 않습니다.)

rows : 세로 크기를 조절하는 속성(화면에 보여지는 줄 수)

- <button> : 버튼을 만들 때 사용하며 submit, reset, button 3가지의 타입이 있습니다.

```
<button type="submit|reset|button">○○○</button>
```

-> 각 버튼은 이전에 배웠던 input 타입의 submit, reset, button과 모두 같은 기능을 가진 버튼입니다.

다만, 빈 태그가 아니며 내용을 안에 직접 넣을 수 있으므로 좀 더 자유로운 스타일 표현이 가능합니다.

13) 폼 요소 4

- <label> : <label>은 form 요소의 이름과 form 요소를 명시적으로 연결시켜주기 위해 사용합니다. 모든 form 요소에 사용할 수 있습니다.

```
<label for="name">이름</label>: <input type="text" id="name"><br>
```

```
<label for="nickname">이름</label>: <input type="text" id="nickname"><br>
```

```
<label for="address">이름</label>: <input type="text" id="address"><br>
```

-> form 요소의 id 속성값과 <label>의 for 속성값을 같게 적어주어야 합니다. <label>을 사용하면 이를 클릭했을 경우 해당 form 요소를 클릭한 것처럼 동작합니다. 또한, 스크린 리더기를 통해 듣게 되면 해당 form 요소에 접근 시 <label>을 함께 읽어주게 됩니다. <label>은 사용성, 접근성적인 측면으로 중요한 역할을 하므로 반드시 써주는 것이 좋습니다.

- <fieldset>, <legend> : <fieldset>, <legend>는 form 요소를 구조화 하기 위해 필요한 태그입니다.

<fieldset> : 여러 개의 폼 요소를 그룹화하여 구조적으로 만들기 위해 사용

<legend> : 폼 요소의 제목으로 <fieldset> 내부에 작성

<fieldset>은 보통 form의 성격에 따라 구분합니다. <legend>는 <fieldset>의 자식으로 반드시 최상단에 위치해야 합니다.

<fieldset>

 <legend>기본 정보</legend>

 ... 폼 요소들 ...

</fieldset>

<fieldset>

 <legend>부가 정보</legend>

 ... 폼 요소들 ...

</fieldset>

- <form> : <form>은 form 요소들을 감싸는 태그로 데이터를 묶어서 실제 서버로 전송해주는 역할을 하는 태그입니다. 만약 <fieldset>으로 구조화되어있다면 <fieldset>도 함께 감싸는 역할을 합니다.

<form action="" method="">

 <fieldset>

 <legend>기본 정보</legend>

 ... 폼 요소들 ...

 </fieldset>

 <fieldset>

 <legend>부가 정보</legend>

 ... 폼 요소들 ...

 </fieldset>

</form>

action: 데이터를 처리하기 위한 서버의 주소

method: 데이터를 전송하는 방식을 지정

method 속성값에는 get/post 2가지 방식이 존재합니다. get 방식은 데이터가 전송될 때 주소창에 파라미터 형태로 붙어 데이터가 노출됩니다. 반면, post 방식은 데이터가 전송될 때 데이터가 노출되지 않습니다.

- 코드 실습

<!DOCTYPE html>

<html lang="ko">

<head>

 <meta charset="UTF-8">

 <title>form</title>

</head>

<body>

 <form action="">

 <h1>Form 관련 요소</h1>

 <fieldset>

 <legend>기본 정보</legend>

 <label for="userid">아이디 : </label> <input type="text" placeholder="영문으로만 써주세요" id="userid">

 <label for="userpw">비밀번호 : </label> <input type="password"

```

id="userpw"><br>
        성별 : <label for="male">남자</label> <input type="radio" name="gender"
id="male" checked>, <label for="female">여자</label> <input type="radio" name="gender"
id="female"><br>
        </fieldset>
        <fieldset>
            <legend>부가 정보</legend>
            취미 : 영화 감상 <input type="checkbox" name="hobby" checked>, 음악 감
상 <input type="checkbox" name="hobby">, 독서 <input type="checkbox" name="hobby"><br>
            프로필 사진 업로드 : <input type="file"><br>
            사는 지역 : <select>
                <option>서울</option>
                <option>경기</option>
                <option>강원</option>
                <option selected>제주</option>
            </select><br>
            자기소개 : <textarea cols="30" rows="5" placeholder="자기소개는 짧게 해주
세요."></textarea><br>
            <button type="submit">전송</button>
            <button type="reset">취소</button>
        </fieldset>
    </form>
</body>
</html>

```

3. 콘텐츠 모델, 시멘틱 마크업, 블록&인라인

1) 콘텐츠 모델

- Content Models의 7 분류

1. Metadata Content
2. Flow Content
3. Sectioning Content
4. Heading Content
5. Phrasing Content
6. Embedded Content
7. Interactive Content

- Metadata : " base, link, meta, noscript, script, style, title " Metadata에는 콘텐츠의 스타일, 동작을 설정하거나 다른 문서와의 관계 등 정보를 포함하는 요소들이 포함됩니다. 메타 태그, 타이틀 태그, 스타일 태그, 링크 태그가 이에 해당하며 대부분 <head>내에 들어간다는 것이 특징입니다.

- Flow : " a, abbr, address, map>area, article, aside, audio, b, bdo, blockquote, br, button, canvas, cite, code, datalist, del, details, dfn, div, dl, em, embed, fieldset, figure, footer, form, h1 ~ h6, header, hgroup, hr, i, iframe, img, input, ins, kbd, keygen, label, map, mark, math, menu, meter, nav, noscript, object, ol, output, p, pre, progress, q, ruby, samp, script, section, select, small, span, strong, style[scoped], sub, sup, svg, table, textarea, time, ul, var, video, wbr " Flow에는 문서의 자연스러운 흐름에 의해 배치되는 요소들이 포함됩니다. Metadata에 해당하는 일부 태그들만 Flow에서 제외되며 요소 대부분이 Flow에 포함됩니다.

- Sectioning : " article, aside, nav, section " Sectioning에는 문서의 구조와 관련된 요소들이 포함됩니다. HTML5에서 새로 생긴 <article>, <aside>, <nav>, <section> 등이 포함되며 이 태그들은 문서의 구조, 아웃라인에 영향을 주게 됩니다.

- Heading : " h1, h2, h3, h4, h5, h6 " Heading에는 각 section의 header를 정의하는 heading 태그가 포함됩니다.

- Phrasing : "a, abbr, map>area, audio, b, bdo, br, button, canvas, cite, code, datalist, del, dfn, em, embed, i, iframe, img, input, ins, kbd, keygen, label, map, mark, math, meter, noscript, object, output, progress, q, ruby, samp, script, select, small, span, strong, sub, sup, svg, textarea, time, var, video, wbr" Phrasing에는 문서의 텍스트 또는 텍스트를 꾸며주는 문단 내부 레벨로 사용되는 요소들이 포함됩니다.

- Embedded : " audio, canvas, embed, iframe, img, math, object, svg, video " Embedded에는 외부 콘텐츠를 표현하는 요소들이 포함되며 오디오나 비디오, 이미지 등 멀티미디어 관련 요소들이 이에 해당합니다.

- Interactive : " a, audio[controls], button, details, embed, iframe, img[usemap], input, keygen, label, menu, object[usemap], select, textarea, video[controls] " Interactive에는 사용자와 상호작용을 하는 요소들이 포함되며 대표적으로 form 요소들이 이에 해당합니다.

https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Content_categories

2) 시멘틱 마크업

- 시멘틱 마크업이란? 시멘틱 마크업은 종종 POSH(Plain Old Semantic HTML)라고도 불리는데, 단어 그대로 평범하고 오래된 의미론적인 HTML이라는 뜻입니다. 시멘틱은 즉, 기계(컴퓨터, 브라우저)가 잘 이해할 수 있도록 하는 것을 뜻합니다. 애초에 프로그래밍 언어는 사람과 기계와의 정해진 약속이며 HTML 역시 마찬가지입니다. 시멘틱 마크업은 적절한 HTML 요소를 올바르게 사용하는 것에서 시작합니다.

- 시멘틱 마크업 하기 : 마크업 할 때는 의미에 맞는 태그, 요소를 사용하는 것이고 문서를 표현할 때는

구조화를 잘 해주는 것입니다. 정해진 약속대로 코드를 작성하게 되면 결국 기계뿐 아니라 사람도 이해하기 쉬운 코드가 됩니다.

굵은 vs 중요한

<i>기울어진</i> vs 강조하는

<u>밑줄친</u> vs <ins>새롭게 추가된</ins>

<s>중간선이 있는</s> vs 삭제된

는 의미 없이 단순히 텍스트를 굵게 표현하는 태그지만, 은 중요하다는 의미를 지닙니다. 은 중요하다는 의미에 맞춰 브라우저에 의해 굵은 스타일로 표현된 것입니다. 따라서 중요하다는 의미를 포함할 때는 가 아닌 을 사용하는 것이 적절하고 시멘틱한 마크업입니다. 이외에 <i>는 단순히 기울어진 글자를 표현하지만, 은 글자의 특정 부분을 강조하는 의미를 지닙니다. <u>와 <s>는 단순히 글자에 선을 그은 것이고, <ins>와 은 내용이 새롭게 추가되거나 삭제가 되었다는 의미를 지닙니다.

<https://developer.mozilla.org/en-US/docs/Glossary/Semantics>

https://www.w3schools.com/html/html5_semantic_elements.asp

<https://nuli.navercorp.com/seminar/s03th>

3) HTML5 시멘틱 요소 : <article> <aside> <figcaption> <figure> <footer> <header> <main> <mark> <nav> <section> <time>

<https://developer.mozilla.org/en-US/docs/Glossary/Semantics>

4) 블록&인라인

- 블록 레벨 요소 : 부모 요소의 가로 영역에 맞게 꽉 채워져 표현되는 요소입니다. 양옆으로 다른 요소가 배치되지 않게 박스를 생성하므로 박스의 위아래로 줄 바꿈이 생기게 됩니다. 블록 레벨 요소는 일반적인 모든 요소(블록, 인라인 레벨 등)를 포함할 수 있습니다. " div, h1~h6, p, ul, li, table ..."

https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements

- 인라인 레벨 요소 : 하나의 라인 안에서 자신의 내용만큼의 박스를 만드는 요소입니다. 라인의 흐름을 끊지 않고 요소 앞 뒤로도 줄 바꿈이 되지 않아 다른 인라인 요소들이 자리할 수 있습니다. 인라인 레벨 요소는 블록 레벨 요소의 자식으로 분류되기 때문에 자손으로 블록 레벨 요소를 가질 수 없습니다. 즉, 인라인 레벨 요소는 블록 레벨 요소를 포함할 수 없습니다. " span, i, img, em, strong, a ..." 다만, HTML5 버전에서 생겨난 한 가지 예외 경우가 있는데 <a>는 인라인 레벨 요소지만 자손으로 블록 레벨 요소를 가질 수 있습니다.

https://developer.mozilla.org/en-US/docs/Web/HTML/Inline_elements

4. CSS 이해하기

1) CSS 소개

- CSS와 HTML : CSS는 간단히 이야기하면 HTML(마크업 언어)을 꾸며주는 표현용 언어입니다. HTML 이 문서의 정보, 구조를 설계한다면 CSS는 문서를 보기 좋게 디자인합니다. CSS는 분명히 HTML과는 독립된 다른 언어지만 마크업 문서 자체가 존재하지 않으면 CSS는 무용지물이나 마찬가지입니다. 왜냐하면, CSS는 표현을 위한 언어인데 마크업 문서가 없다면 표현할 대상이 없기 때문입니다. 그래서 CSS는 보통 마크업 언어인 HTML과 같이 묶어서 이야기합니다.

- CSS로 표현한 다양한 웹 사이트들 : 전 세계에 많은 웹 사이트들이 있습니다. 그리고 그것들 모두 HTML 태그를 이용해서 만들어졌고, 자주 사용되는 태그의 개수는 10여 개밖에 되지 않습니다. 결국, 모든 사이트가 비슷한 HTML 태그를 사용해서 만들어졌음에도 불구하고 각각 다양하고 독창적인 디자인으로 표현 가능한 이유가 바로 CSS 덕분입니다. CSS는 문서를 디자인하는 강력한 힘을 가졌습니다.

<http://www.csszengarden.com/>

2) CSS 문법과 적용

- CSS 문법 : `h1{color: yellow; font-size: 2em;}`

CSS는 HTML 요소를 꾸며주는 역할을 합니다.

CSS는 꾸밀 대상이 되는 요소와 그에 대한 스타일 내용으로 이루어져 있습니다.

- CSS 구문

선택자(selector) : "h1"

속성(property) : "color"

값(value) : "yellow"

선언(declaration) : "color: yellow", "font-size: 2em"

선언부((declaration block) : "{ color: yellow; font-size:2em; }"

규칙(rule set) : "h1 { color: yellow; font-size:2em; }"

CSS 파일은 여러 개의 규칙으로 이루어져 있습니다. 선택자와 선언부 사이, 선언과 선언 사이는 앞뒤로 개행을 해도 상관이 없습니다. 하지만 속성이름과 속성값 사이에는 개행을 하면 안 됩니다.

[올바른 CSS]

h1

```
{ color: yellow; font-size:2em; }
```

h1 {

color: yellow;

font-size:2em;

}

[잘못된 CSS]

h1 {

color:

yellow;

}

- CSS의 속성(Property)과 HTML의 속성(Attribute) : HTML에도 속성이 있고, CSS에도 속성이 있습니다. 두 가지는 전혀 다른 것입니다. HTML의 속성은 영어로 attribute이고, CSS의 속성은 property입니다. 둘 다 한국어로 번역할 때 "속성"이라고 하므로 잘 구분하셔야 합니다.

- CSS 주석

/* 주석 내용 */

/*

주석은 여러 줄로도
선언 할 수 있습니다.

*/

- CSS의 적용

1. Inline : Inline은 해당 요소에 직접 스타일 속성을 이용해서 규칙들을 선언하는 방법입니다. 해당 요소에 직접 입력하기 때문에 선택자는 필요하지 않게 되고, 선언부에 내용만 스타일 속성의 값으로 넣어주면 됩니다. Inline 스타일 방식이라고 부릅니다.

```
<div style="color:red;"> 내용 </div>
```

Inline 스타일 방식은 코드의 재활용이 되지 않기 때문에 자주 사용하지 않습니다.

2. Internal : Internal은 문서에 <style>을 활용한 방법입니다. <style>은 <head>내부에 들어가며 <style>안에 스타일 규칙이 들어갑니다.

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

위의 코드로 모든 <div>에 같은 스타일을 줄 수 있습니다. 하지만 이것도 한계가 있습니다. 많은 페이지가 있는 경우에는 모든 페이지에 저마다의 규칙을 선언해줘야 합니다. 페이지가 많고 스타일 규칙 내용이 많아지면 결코 쉬운 일은 아닙니다.

3. External : External은 외부 스타일 시트 파일을 이용한 방법입니다. 외부 스타일 시트는 스타일 규칙들을 별도의 외부 파일을 만들어 넣는 방식입니다. 외부 파일은 확장자가 .css가 되며 css 파일이라고 부릅니다.

```
div {color: red;}
```

우선 CSS 파일을 하나 만들고 스타일 규칙을 선언합니다. 그다음 <link>을 이용해서 CSS 파일을 연결하면 됩니다.

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

<head> 내부에 <link>를 선언한 후 href 속성을 이용해 CSS 파일의 경로를 적습니다. rel 속성은 연결되는 파일이 문서와 어떤 관계인지를 명시하는 속성으로, CSS 파일은 'stylesheet' 라고 적어야 합니다. 외부 스타일 시트 방식으로 스타일을 선언하면 많은 페이지가 있더라도 이 한 줄로 모든 페이지에 같은 스타일을 적용할 수 있습니다. 또한, 수정이 필요할 때도 CSS 파일을 수정하면 연결된 모든 페이지에 반영할 수 있습니다. 외부 스타일 시트 방식은 파일 관리가 편하면서도 용량이 작기 때문에 주로 사용되는 방법입니다.

4. Import : Import는 스타일 시트 내에서 다른 스타일 시트 파일을 불러오는 방식입니다.

```
@import url("css/style.css");
```

<style> 내부 상단이나 외부 스타일 시트 파일 상단에 선언하는데 성능상 좋지 않아서 거의 쓰이지 않습니다. 이에 본 강의에서는 다루지 않습니다. 궁금하신 분들은 따로 찾아보시길 바랍니다.

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>css</title>
```

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

```
</head>
```

```
<body>
```

```
  <p>Hello, CSS</p>
```

```
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Culpa debitis facere fuga
```

laboriosam placeat provident, quibusdam quidem quo sapiente totam?</p>
</body>
</html>

https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/How_CSS_is_structured
https://www.w3schools.com/css/css_howto.asp

3) 선택자1

- 요소 선택자: 선택자 중에 가장 기본이 되는 선택자이며, 태그 선택자라고도 합니다.

```
h1 { color: yellow; }  
h2 { color: yellow; }  
h3 { color: yellow; }  
h4 { color: yellow; }  
h5 { color: yellow; }  
h6 { color: yellow; }
```

요소 선택자는 위 코드처럼 선택자 부분에 태그 이름이 들어갑니다. 문서 내에 선택자에 해당하는 모든 요소에 스타일 규칙이 적용됩니다.

- 그룹화 : 선택자는 쉼표를 이용해서 그룹화를 할 수 있습니다.

```
h1, h2, h3, h4, h5, h6 { color: yellow; }
```

위 코드는 요소 선택자의 예제 코드와 같은 코드입니다. 그리고 전체 선택자 라고 불리는 간단한 선택자도 있습니다.

```
* { color: yellow; }
```

*(별표, asterisk) 기호로 문서 내에 모든 요소를 선택할 수 있습니다. 이렇게 선언하면, 한 번의 선언만으로 문서 내에 모든 요소에 스타일 규칙이 적용됩니다. 전체 선택자는 매우 편리하지만, 성능에 좋지 않으므로 될 수 있으면 사용을 지양합니다. 선택자뿐만 아니라 선언들도 그룹화가 가능합니다.

```
h1 { color: yellow; font-size: 2em; background-color: gray; }
```

그리고 마지막으로 선택자와 선언이 동시에 그룹화도 가능합니다.

```
h1, h2, h3, h4, h5, h6 { color: yellow; font-size: 2em; background-color: gray; }
```

<코드실습 >

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <title>css</title>
```

```
    <style>
```

```
        h1 { color: yellow; font-size: 2em; background-color: gray; }
```

```
    </style>
```

```
</head>
```

```
<body>
```

```
    <h1>Hello, CSS</h1>
```

```
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Culpa debitis facere fuga laboriosam placeat provident, quibusdam quidem quo sapiente totam?</p>
```

```
</body>
```

```
</html>
```

https://developer.mozilla.org/en-US/docs/Glossary/CSS_Selector

4) 선택자2

- class 선택자 : 요소에 구애받지 않고 스타일 규칙을 적용할 수 있는 가장 일반적인 방법은 class 선택자를 활용하는 것입니다. class 선택자를 사용하기 위해서는 HTML을 수정해 class 속성을 추가해야 합니다. class 속성은 글로벌 속성이므로 어느 태그에서도 사용할 수 있습니다. class 속성에 값을 넣게 되면, class 선택자를 이용해서 해당 요소에 스타일 규칙을 적용할 수 있습니다.

```
.foo { font-size: 30px; }
```

```
<p class="foo"> ... </p>
```

<p>의 class 속성의 값으로 "foo"라는 값을 넣었다면, CSS에서 그 값("foo")을 선택자로 지정하면 됩니다. 클래스 선택자를 쓸 때는, 맨 앞에 .(마침표)를 찍어주셔야 합니다. 이렇게 되면 어느 요소든지 class 속성값이 "foo"로 선언된 요소가 있다면 해당 스타일 규칙을 적용받게 됩니다. 그럼 이제 클래스 선택자를 이용해서 인트로에 제시했던 퀴즈를 풀어보겠습니다.

```
.html { color: purple; }
```

```
.css { text-decoration: underline; }
```

```
<dl>
```

```
    <dt class="html">HTML</dt>
```

```
    <dd><span class="html">HTML</span>은 문서의 구조를 나타냅니다.</dd>
```

```
    <dt class="css">CSS</dt>
```

```
    <dd><span class="css">CSS</span>는 문서의 표현을 나타냅니다.</dd>
```

```
</dl>
```

- 다중 class : class 속성은 꼭 하나의 값만 가질 수 있는 것은 아닙니다. 공백으로 구분하여 여러 개의 class 값을 넣을 수 있습니다.

```
.foo { font-size: 30px; }
```

```
.bar { color: blue; }
```

```
<p class="foo bar"> ... </p>
```

<p>에 class 속성에 "foo" 와 "bar" 2개의 값을 넣었습니다. 그리고 foo class 선택자에는 폰트의 크기를 30px로, bar class 선택자에는 글자를 파란색으로 적용하는 스타일 규칙이 선언되어있었습니다. 그렇게 되면 이 <p>에는 2개의 규칙이 모두 적용이 됩니다.

- id 선택자 : id 선택자는 class 선택자와 비슷합니다. 선택자를 쓸 때는, .(마침표) 기호 대신 #(해시) 기호를 써주시면 되고, 요소에는 class 속성 대신 id 속성만 써주면 됩니다.

```
#bar { background-color: yellow; }
```

```
<p id="bar"> ... </p>
```

이 <p>는 id 선택자의 스타일 규칙이 적용됩니다.

- class 선택자와의 차이점 : 기호가 아닌 #기호 사용, 태그의 class 속성이 아닌 id 속성을 참조, 문서 내에 유일한 요소에 사용, 구체성

가장 큰 차이점은 class와 달리 id는 문서 내에서 유일해야 한다는 점입니다. 클래스 선택자는 여러 요소에 같은 클래스를 넣고 같은 규칙을 적용할 수 있었습니다. 그리고 그것이 클래스 선택자의 장점이기도 합니다. 하지만 id 속성값은 문서 내에 유일하게 사용이 되어야 합니다. 결국, id 선택자로 규칙을 적용할 수 있는 요소는 단 하나뿐입니다. 그리고 마지막으로 구체성의 값이 다릅니다.

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```

<meta charset="UTF-8">
<title>css</title>
<style>
    .item { background: gray; }
    .a { color: yellow; }
    .b { color: blue; }
    #special { color: red; }
</style>
</head>
<body>
    <ul>
        <li class="item a">first</a>
        <li class="item b">second</a>
        <li class="item" id="special">third</a>
    </ul>
</body>
</html>
https://css-tricks.com/the-difference-between-id-and-class/

```

5) 선택자 3

- 선택자의 조합

/* 요소와 class의 조합 */

```
p.bar { ... }
```

/* 다중 class */

```
.foo.bar { ... }
```

/* id와 class의 조합 */

```
#foo.bar { ... }
```

첫 번째는 요소와 클래스를 조합한 경우입니다. 이 경우에는 <p>이면서 class 속성에 bar가 있어야 적용됩니다.

두 번째는 다중 클래스의 경우입니다. 이 경우에는 class 속성에 foo와 bar가 모두 있어야 적용됩니다.

마지막은 id와 class를 조합한 경우입니다. 이 경우에는 id가 foo이며 class가 bar인 요소에 적용됩니다.

- 속성 선택자 : 단순 속성으로 선택

```
p[class] { color: silver; }
```

```
p[class][id] { text-decoration: underline; }
```

```
<p class="foo">Hello</p>
```

```
<p class="bar">CSS</p>
```

```
<p class="baz" id="title">HTML</p>
```

속성 선택자는 대괄호를 이용해서 선언하며 대괄호 안에 속성 이름이 들어갑니다. 요소에 해당 이름의 속성이 있다면 해당 사항이 적용됩니다. 위 CSS 코드는 요소 선택자와의 조합으로 이루어진 코드입니다. 첫 번째는 <p>이면서 class 속성이 있는 요소이면 color: silver 규칙이 적용됩니다. 두 번째는 <p>이면서 class 속성과 id 속성이 함께 있어야 text-decoration: underline 규칙이 적용됩니다. 바로 위 HTML 코드에는 3개의 <p>가 있습니다. 그렇다면 이 3개의 <p>에는 각자 어떤 스타일이 적용될까요? 먼저 예측

을 하시고 직접 실습을 하는 게 좋습니다. p[class] 선택자의 규칙은 class 속성만 존재하면 적용이 되기 때문에 3가지 요소 모두에 적용됩니다. p[class][id] 선택자의 규칙은 class 속성과 id 속성 모두 있는 요소만 해당하기 때문에 마지막 요소에만 적용됩니다. 두 규칙 모두 속성의 값은 상관하지 않습니다.

: 정확한 속성값으로 선택

정확한 속성값으로 선택은 제목 그대로 속성의 값으로 요소를 선택합니다. 선택자는 대괄호 안에 속성 이름과 속성값을 다 적으면 됩니다.

```
p[class="foo"] { color: silver; }
```

```
p[id="title"] { text-decoration: underline; }
```

p[class="foo"]는 <p>이면서 class 속성의 값이 foo이면 적용되고, p[id="title"]는 <p> 이면서 id 속성의 값이 title이면 적용됩니다.

: 부분 속성값으로 선택

부분 속성값으로 선택은 속성 이름과 속성값 사이에 사용되는 기호에 따라 동작이 조금 다릅니다.

[class~="bar"] : class 속성의 값이 공백으로 구분한 "bar" 단어가 포함되는 요소 선택

[class^="bar"] : class 속성의 값이 "bar"로 시작하는 요소 선택

[class\$="bar"] : class 속성의 값이 "bar"로 끝나는 요소 선택

[class*="bar"] : class 속성의 값이 "bar" 문자가 포함되는 요소 선택

```
<p class="color hot">red</p>
```

```
<p class="cool color">blue</p>
```

```
<p class="colorful nature">rainbow</p>
```

```
p[class~="color"] { font-style: italic; }
```

```
p[class^="color"] { font-style: italic; }
```

```
p[class$="color"] { font-style: italic; }
```

```
p[class*="color"] { font-style: italic; }
```

위의 코드에서는 모두 class 속성값으로 "color"를 선택합니다. 요소 순서대로 기호에 따라 규칙이 적용되는 결과는 다음과 같습니다.

```
p[class~="color"] { font-style: italic; } /* 1, 2번째 요소 */
```

```
p[class^="color"] { font-style: italic; } /* 1, 3번째 요소 */
```

```
p[class$="color"] { font-style: italic; } /* 2번째 요소 */
```

```
p[class*="color"] { font-style: italic; } /* 1, 2, 3번째 요소 */
```

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <title>css</title>
```

```
    <style>
```

```
        p[class$="color"] { font-style: italic; }
```

```
    </style>
```

```
</head>
```

```
<body>
```

```
    <p class="color hot">red</p>
```

```
    <p class="cool color">blue</p>
```

```
    <p class="colorful nature">rainbow</p>
```

```
</body>
```

</html>

https://www.w3schools.com/css/css_attribute_selectors.asp

https://developer.mozilla.org/ko/docs/Web/CSS/Attribute_selectors

6) 문서 구조 관련 선택자

- 문서 구조의 이해 : 선택자와 문서의 관계를 이해하기 위해 먼저 어떻게 문서가 구조화되는지를 다시 한 번 살펴보겠습니다.

<html>

<body>

<div>

<h1>HTML: Hyper Text Markup Language</h1>

</div>

<p>HTML과 CSS와 JAVASCRIPT를 이용해서 멋진 웹 사이트를 제작할 수 있습니다.</p>

</body>

</html>

부모와 자식 : 부모 요소는 그 요소를 포함하는 가장 가까운 상위 요소로, 그 요소의 부모 요소는 단 하나뿐입니다. 자식 요소는 부모 요소와 반대라고 생각하면 되며 자식 요소는 여러 개일 수도 있습니다.

<body>의 부모 요소: <html> ↔ <html>의 자식 요소: <body>

<div>의 부모 요소: <body> ↔ <body>의 자식 요소: <div>, <p>

<h1>의 부모 요소: <div> ↔ <div>의 자식 요소: <h1>

의 부모 요소: <h1> ↔ <h1>의 자식 요소:

<p>의 부모 요소: <body> ↔ <body>의 자식 요소: <div>, <p>

조상과 자손 : 조상과 자손의 관계는 부모와 자식의 관계와 비슷합니다. 정확히 얘기하면 부모와 자식의 관계를 포함한 확장된 관계라고 생각하면 됩니다. 조상 요소는 그 요소를 포함하는 모든 요소로, 부모 요소를 포함하여 여러 개일 수도 있습니다. 자손 요소는 그 반대로, 그 요소가 포함하고 있는 모든 요소가 자손 요소입니다.

<body>의 조상 요소: <html> ↔ <html>의 자손 요소: <body>, <div>, <h1>, , <p>

<div>의 조상 요소: <html>, <body> ↔ <body>의 자손 요소: <div>, <h1>, , <p>

<h1>의 조상 요소: <html>, <body>, <div> ↔ <div>의 자손 요소: <h1>,

의 조상 요소: <html>, <body>, <div>, <h1> ↔ <h1>의 자손 요소:

<p>의 조상 요소: <html>, <body> ↔ <body>의 자손 요소: <div>, <h1>, , <p>

보통 문서의 요소들은 모두 이처럼 어느 요소의 자식(자손) 요소이자 부모(조상) 요소가 되는 경우가 많습니다.

형제 : 부모와 자식, 조상과 자손 말고도 형제 관계도 있습니다. 같은 부모를 가지고 있는 요소들은 서로 형제 관계에 있습니다. 위 코드에서는 <div>, <p>가 형제 요소입니다. 형제 관계 중에는 인접한 관계도 있습니다. 형제 관계에 있는 요소 중 바로 뒤에 이어 나오는 요소를 인접해 있다고 합니다. 여기서 <p>가 <div>에 인접한 형제 요소가 됩니다. 문서의 구조는 흔히 가계도나 조직도의 관계와 비슷하다고 생각하면 이해하기가 쉽습니다.

- 문서 구조 관련 선택자 : 문서 구조를 이용한 선택자는 3가지 있습니다.

자손 선택자 : 자손 선택자는 선택자 사이에 아무 기호 없이 그냥 공백으로 구분을 합니다.

div span { color: red; } 이 선택자는 <div>의 자손 요소인 를 선택하는 선택자 입니다.

자식 선택자 : 자식 선택자는 선택자 사이에 닫는 꺾쇠 기호(>)를 넣습니다. 꺾쇠 기호와 선택자 기호 사이에는 공백은 있거나 없어도 상관 없습니다.

div > h1 { color: red; } 이 선택자는 <div>의 자식 요소인 <h1>를 선택하는 선택자입니다.

인접 형제 선택자 : div + p { color: red; }

인접 형제 선택자는 선택자 사이에 + 기호를 넣습니다. 자식 선택자와 마찬가지로 공백은 있거나 없어도 상관 없습니다. 인접 형제 선택자는 형제 관계이면서 바로 뒤에 인접해 있는 요소를 선택하는 선택자입니다.

/* body 요소의 자식인 div 요소의 자손인 table 요소 바로 뒤에 인접한 ul 요소 선택! */

body > div table + ul { ... }

문서 구조 관련 선택자는 더 복잡하게 사용할 수 있습니다. 유의할 점은 요소들이 많이 나열되어 있더라도 제일 우측에 있는 요소가 실제 선택되는 요소라는 것입니다.

https://www.w3schools.com/cssref/sel_gen_sibling.asp

https://developer.mozilla.org/en-US/docs/Web/CSS/Adjacent_sibling_combinator

https://developer.mozilla.org/ko/docs/Web/CSS/General_sibling_selectors

https://developer.mozilla.org/en-US/docs/Web/CSS/Child_combinator

https://developer.mozilla.org/en-US/docs/Web/CSS/Descendant_combinator

https://www.w3schools.com/cssref/sel_gen_sibling.asp

7) 가상 선택자 1

- 가상 클래스(pseudo class) : 가상 클래스는 미리 정의해놓은 상황에 적용되도록 약속된 보이지 않는 클래스입니다. 우리가 직접 요소에 클래스를 선언하는 것은 아니고, 약속된 상황이 되면 브라우저 스스로 클래스를 적용해줍니다.

가상 클래스가 없다면 이런 과정을 거치게 됩니다.

1. 임의의 클래스 선택자를 선언해 특정 스타일 규칙을 만든다.
2. p 요소에 커서 올라가면 p 요소에 클래스를 집어넣는다.
3. p 요소에서 커서가 빠지면 p 요소에 클래스를 삭제한다.

여기서 2, 3번은 동적으로 변화되어야 하는데, HTML과 CSS는 정적인 언어이기 때문에 처리할 수 없습니다. 어쩔 수 없이 다른 언어를 사용해야 하는데, 이는 개발 비용이 들어가는 일입니다. 그래서 CSS에서는 흔하게 사용되는 패턴에 대해서 미리 정의해놓고, 가상 클래스로 제어할 수 있게 했습니다.

```
:pseudo-class {  
    property: value;  
}
```

가상 클래스는 :(콜론) 기호를 써서 나타냅니다. 가상 클래스를 이용하면 아래의 경우에도 CSS만으로 처리가 가능하므로 훨씬 효율적입니다. ":hover 가상 클래스 선택자를 이용해서 스타일 규칙을 만든다. (hover는 마우스 커서가 올라갔을 때 적용이 되도록 정의되어 있습니다.)" 가상 클래스에는 여러 가지가 있습니다. <https://developer.mozilla.org/ko/docs/Web/CSS/Pseudo-classes>

- 문서 구조와 관련된 가상 클래스 : 문서 구조와 관련된 가상 클래스는 first-child와 last-child 가상 클래스 선택자입니다.

:first-child : 첫 번째 자식 요소 선택

:last-child : 마지막 자식 요소 선택

```
<ul>  
    <li>HTML</li>  
    <li>CSS</li>  
    <li>JS</li>
```

```
</ul>
```

```
li:first-child { color: red; }
```



```
li:last-child { color: blue; }
```

첫 번째 와 마지막 에 가상 클래스가 적용됩니다. 실제 에는 class 속성이 없지만 내부적으로 가상 클래스가 적용되어 마치 아래의 코드와 같이 동작하게 됩니다.

```
<ul>
  <li class="first-child">HTML</li>
  <li>CSS</li>
  <li class="last-child">JS</li>
</ul>
```

- 앵커 요소와 관련된 가상 클래스 : 앵커 요소와 관련된 가상 클래스로는 :link와 :visited가 있습니다.

:link : 하이퍼 링크이면서 아직 방문하지 않은 앵커

:visited : 이미 방문한 하이퍼링크를 의미

하이퍼 링크는 앵커 요소에 href 속성이 있는 것을 의미합니다.

```
a:link { color: blue; }
```

```
a:visited { color: gray; }
```

- 사용자 동작과 관련된 가상 클래스 : 이 클래스들도 <a>에 주로 많이 쓰입니다. <a>에만 쓸 수 있는 것은 아니며, 이 조건에 맞는 상황이 되는요소들을 다 사용이 가능합니다.

:focus: 현재 입력 초점을 가진 요소에 적용

:hover: 마우스 포인터가 있는 요소에 적용

:active: 사용자 입력으로 활성화된 요소에 적용

```
a:focus { background-color: yellow; }
```

```
a:hover { font-weight: bold; }
```

```
a:active { color: red; }
```

:focus는 현재 입력 초점을 가진 요소에 적용됩니다.

focus(초점)는 지금 현재 선택을 받는 것을 의미합니다. 예를 들면, 입력 폼 요소에 텍스트를 입력하려고 마우스 클릭해서 커서를 입력 폼 위에 올려놓으면 그때 입력 폼 요소가 초점을 받는 상태입니다. 또 키보드의 탭 키를 이용해서 요소를 탐색하다 보면 링크나 버튼에 점선 테두리가 이동하는 것을 볼 수 있는데, 점선 테두리가 위치하는 것도 초점을 받은 상태입니다.

:hover는 마우스 커서가 있는 요소에 적용됩니다. (마우스를 올렸을 때를 의미합니다.)

:active는 사용자 입력으로 활성화된 요소를 의미하는데, <a>를 클릭할 때 또는 <button>를 눌렀을 때처럼 순간적으로 활성화됩니다.

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>css</title>
```

```
  <style>
```

```
    a:focus { background-color: yellow }
```

```
    a:hover { font-weight: bold }
```

```
    a:active { color: red }
```

```
  </style>
```

```
</head>
```

```
<body>
```

```
  <a href="http://www.naver.com">네이버</a>
```

```

        <a href="http://www.google.com">구글</a>
        <a href="http://www.daum.net">다음</a>
</body>
</html>
https://www.w3schools.com/css/css\_pseudo\_classes.asp

```

8) 가상 선택자 2

- 가상 요소(pseudo element) : 가상 요소는 HTML 코드에 존재하지 않는 구조에 스타일을 부여할 수가 있습니다. 가상 요소도 가상 클래스처럼 문서 내에 보이지 않지만, 미리 정의한 위치에 삽입되도록 약속이 되어있습니다. 선언 방법은 가상 클래스와 같이 콜론을 사용하며, CSS3부터는 가상 클래스와 가상 요소를 구분하기 위해 가상 요소에는 ::(더블 콜론) 기호를 사용하기로 했습니다. 하지만, 하위 브라우저에서 ::문법을 지원하지 않는 문제가 있으므로 상황에 따라 : 기호를 사용하셔야 합니다.

```

::pseudo-element {
    property: value;
}

```

<https://developer.mozilla.org/ko/docs/Web/CSS/Pseudo-elements>

:before : 가장 앞에 요소를 삽입

:after : 가장 뒤에 요소를 삽입

:first-line : 요소의 첫 번째 줄에 있는 텍스트

:first-letter : 블록 레벨 요소의 첫 번째 문자

```

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>

```

```

p::before { content: "###" }

```

```

p::after { content: "!!!" }

```

```

p::first-line { ... }

```

```

p::first-letter { ... }

```

before와 after 가상 요소는 애초에 내용이 없는 상태로 생성되기 때문에 내용을 넣기 위해 content 속성을 이용해야 합니다. 실제 HTML 코드에는 나타나지 않지만, before와 after가 어떻게 동작하는지 이해를 돕기 위해 코드를 아래와 같이 변경했습니다.

```

<p>

```

```

    <before>###</before>

```

```

    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

```

```

    <after>!!!</after>

```

```

</p>

```

눈에 보이지 않지만, 내부에서 이처럼 요소가 생성됩니다. first-line과 first-letter도 마찬가지로 아래 코드와 같은 것으로 생각하시면 됩니다.

```

<p>

```

```

    <first-letter>L</first-letter>orem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

```

```

</p>

```

```

<p>

```

```

    <!-- 모니터 가로 해상도에 따라 요소가 포함하는 내용이 변동됩니다. -->

```

```

    <first-line>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiu ..(..어딘가

```

쯤..) </first-line>... unt ut labore et dolore magna aliqua.
</p>

<https://code.tutsplus.com/ko/tutorials/the-30-css-selectors-you-must-memorize-net-16048>

<https://developer.mozilla.org/en-US/docs/Web/CSS/content>

https://www.w3schools.com/cssref/pr_gen_content.asp

9) 구체성

- 구체성 : 요소를 선택하는 데는 여러 방법이 있습니다. 따라서 서로 다른 선택자를 이용해 같은 요소를 선택할 수도 있습니다.

```
h1 { color: red; }
```

```
body h1 { color: green; }
```

서로 다른 규칙들이 상반된 스타일을 가지고 있다. 두 규칙은 모두 <h1>을 선택하게 됩니다. 하지만, 두 규칙이 지정하는 스타일은 서로 다릅니다. <h1>에는 color: green이 적용되는데 이는 구체성과 연관이 있습니다. 선택자에는 어떤 규칙이 우선으로 적용되어야 하는지에 대해 정해진 규칙이 있습니다. 이 규칙을 '구체성'이라고 합니다. 구체성은 선택자를 얼마나 명시적으로(구체적으로) 선언했느냐를 수치화한 것으로, 구체성의 값이 클수록 우선으로 적용이 됩니다.

0, 0, 0, 0

구체성은 4개의 숫자 값으로 이루어져 있습니다. 값을 비교할 때는 좌측에 있는 값부터 비교하며, 좌측 부분의 숫자가 클수록 높은 구체성을 갖습니다. 구체성은 아래의 규칙대로 계산됩니다.

0, 1, 0, 0 : 선택자에 있는 모든 id 속성값

0, 0, 1, 0 : 선택자에 있는 모든 class 속성값, 기타 속성, 가상 클래스

0, 0, 0, 1 : 선택자에 있는 모든 요소, 가상 요소

전체 선택자는 0, 0, 0, 0을 가진다.

조합자는 구체성에 영향을 주지 않는다. (>, + 등)

```
h1 { ... } /* 0,0,0,1 */
```

```
body h1 { ... } /* 0,0,0,2 */
```

```
.grape { ... } /* 0,0,1,0 */
```

```
*.bright { ... } /* 0,0,1,0 */
```

```
p.bright em.dark { ... } /* 0,0,2,2 */
```

```
#page { ... } /* 0,1,0,0 */
```

```
div#page { ... } /* 0,1,0,1 */
```

선택자의 구체성 값을 잘 알아야 많은 스타일 규칙들을 정의할 때 의도하지 않은 일이 생기지 않습니다.

- 인라인 스타일

p#page { color: red; } 0, 1, 0, 1의 구체성을 가지는 선택자로 스타일 선언

<p id="page" style="color:blue">Lorem impusm dolor sit.</p> 요소에 직접 인라인 스타일 방식으로 스타일을 선언

결과적으로 <p>에는 color: blue가 적용됩니다. 인라인 스타일의 구체성 값은 1, 0, 0, 0이며 규칙들 중 가장 큰 구체성을 갖기 때문입니다.

- Important : important 키워드는 별도의 구체성 값은 없지만, 모든 구체성을 무시하고 우선권을 갖습니다. important 키워드는 속성값 뒤 한 칸 공백을 주고 느낌표 기호와 함께 씁니다.

```
p#page { color: red !important; }
```

<p id="page" style="color:blue">Lorem impusm dolor sit.</p>

<p>에는 important로 인해 color: red가 적용됩니다.

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance

10) 상속

- 상속되는 속성

```
h1 { color: gray; }
```

```
<h1>Hello, <em>CSS</em></h1>
```

은 부모인 <h1>의 color: gray를 상속받습니다. 상속은 자연스러운 현상처럼 보이지만, 모든 속성이 다 상속되는 것은 아닙니다. 아직 속성에 대해 다 배우지는 않았지만, margin, padding, background, border 등 박스 모델 속성들은 상속되지 않는다는 것을 알고 계시면 됩니다. 상속되는 속성들은 보통 상식적으로 구분될만한 속성들이며, 후에 속성들에 대해 배우게 되면 자연스럽게 이해할 수 있습니다.

- 상속되는 속성의 구체성

```
* { color: red; }
```

```
h1#page { color: gray; }
```

```
<h1 id="page">Hello, <em>CSS</em></h1>
```

전체 선택자를 이용해 color: red를 적용하고 id 선택자를 이용해 color: gray를 선언했습니다. 전체 선택자의 구체성은 0, 0, 0, 0 이며 id 선택자의 구체성은 0,1,0,1입니다. 에는 color: red가 적용되는데 그 이유는 바로 상속된 속성은 아무런 구체성을 가지지 못하기 때문입니다.

<https://developer.mozilla.org/en-US/docs/Web/CSS/inheritance>

11) 캐스케이딩(cascading)

```
h1 { color: red; }
```

```
h1 { color: blue; }
```

<h1>에는 같은 구체성을 가진 두 규칙이 적용되었습니다. <h1>에는 color: blue가 적용되며 이는 cascading 규칙에 의해 적용된 결과입니다.

- cascading 규칙 :중요도(!important)와 (CSS)출처, 구체성, 선언 순서

CSS 출처는 제작자와 사용자, 그리고 사용자 에이전트(user agent) 경우로 구분합니다. 제작자의 경우는 사이트를 실제 제작하는 개발자가 작성한 CSS를 의미합니다. (대부분이 여기에 해당합니다.) 그리고 사용자의 경우는 웹 페이지를 방문하는 일반 사용자들이 작성한 CSS를 의미합니다. 마지막으로 사용자 에이전트의 경우는 일반 사용자의 환경, 즉 브라우저에 내장된 CSS를 의미합니다. 현재의 브라우저에서는 사용자 스타일을 지원하지 않는 추세이기 때문에 이와 관련해서는 생략하도록 하겠습니다.

모든 스타일은 규칙에 따라 단계적으로 적용된다.

1. 스타일 규칙들을 모아서 중요도가 명시적으로 선언되었는지에 따라 분류합니다.

중요도가 명시적으로 선언된 규칙들은 그렇지 않은 규칙들보다 우선합니다.

중요도가 있는 규칙들끼리는 아래 다른 규칙들을 적용받습니다.

2. 스타일 규칙들을 출처에 따라 분류합니다.

제작자 스타일 규칙이 사용자 에이전트 스타일 규칙보다 우선합니다.

3. 스타일 규칙들을 구체성에 따라 분류합니다.

구체성이 높은 규칙들이 우선합니다.

4. 스타일 규칙들을 선언 순서에 따라 분류합니다.

뒤에 선언된 규칙일수록 우선합니다.

```
<p id="bright">Hello, CSS</p>
```

```
p#bright { color: silver; }
```

```
p { color: red; }
```

구체성에 따라 color: silver가 적용

```
p { color: silver; }
```

```
p { color: red; }
```

선언 순서에 따라 color: red가 적용

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance

<https://developer.mozilla.org/en-US/docs/Web/CSS/Cascade>

12) 선택자 정리

https://www.w3schools.com/cssref/css_selectors.asp

5. 단위, 배경, 박스모델

1) 속성-정의와 구문

- 정의 : 해당 속성이 어떤 변화를 일으키고 어떻게 동작하는지 파악할 수 있습니다. 기본값, 상속 여부, 애니메이션 가능 여부, 사용 가능한 CSS 버전

- 문법 : 해당 속성값을 어떤 식으로 나열해 사용하는지를 파악할 수 있습니다.

- 속성값 : 해당 속성이 인식해 적용할 수 있는 값의 형태나, 키워드 등을 파악할 수 있습니다.

Initial : 초기값, 즉 속성의 기본값으로 정의 (ie에서 지원하지 않음)

Inherit : 부모 요소의 해당 속성값을 적용 (상속 가능할 요소일 경우) 즉, 상속이 불가능한 속성일 경우에는 적용 되지 않습니다.

- 지원 범위 : 해당 속성이 정의에 맞게 동작 가능한 CSS 버전, 브라우저별 버전을 확인할 수 있습니다. 어떤 브라우저의 어떤 버전이나에 따라 같은 값이어도 다르게 렌더링 될 수 있으므로, 현재 프로젝트의 사용자 제공 지원 범위를 잘 확인하고 적용해야 합니다.

- 예제 : 문법과 속성값을 바탕으로 실제로 속성을 동작하는 예제 코드를 확인할 수 있습니다.

- 참고사항 : 해당 속성에 대해 특이사항이나 버그에 대해서 확인할 수 있습니다.

<https://www.w3.org/>

<https://www.w3schools.com/>

<https://developer.mozilla.org/ko/>

2) 속성-단위

- 절대 길이 : 절대 길이는 고정된 크기 단위로, 다른 요소의 크기에 의해 영향을 받지 않습니다.

px (1px = 1/96th of 1 inch)

절대 길이이므로 다른 요소의 영향을 받지 않아 화면에서 고정된 크기를 가지지만, 장치의 해상도에 따라 상대적입니다. 여러 환경에서 디자인을 같게 표현하고 브라우저 호환성에 유리한 구조로 되어 있어서, 디자인 의도가 많이 반영된 웹 사이트의 경우 픽셀 단위를 사용하는 것을 권장하고 있습니다.

pt (1pt = 1/72 of 1 inch)

컴퓨터가 없던 시절부터 있던 단위입니다. 인쇄물이나 워드프로세서 프로그램에서 사용된 가장 작은 표준 인쇄 단위입니다. 웹 화면에 인쇄용 문서를 위한 스타일을 적용할 때 유용하게 사용할 수 있습니다.

그러나 사용하는 기기의 해상도에 따라 차이가 있어 W3C에서도 pt는 웹 개발 시 권장하는 단위가 아닙니다. 예를 들면 Windows에서는 9pt = 12px, Mac에서는 9pt = 9px로 보이게 됩니다.

- 상대 길이 : 상대 길이는 다른 요소의 크기나 폰트 크기, 브라우저(viewport) 등의 크기에 따라 상대적으로 값이 변합니다.

% : 부모의 값에 대해서 백분율로 환산한 크기를 갖게 됩니다.

em : font-size를 기준으로 값을 환산합니다. 소수점 3자리까지 표현 가능합니다.

rem : root의 font-size를 기준으로 값을 환산합니다.

vw : viewport의 width값을 기준으로 1%의 값으로 계산됩니다.

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units

https://www.w3schools.com/cssref/css_units.asp

3) 속성-색상

- Color 속성 : 폰트의 색상 값을 적용할 때 사용하는 속성입니다. h1{color: 색상 값;}

- 색상 값 지정 방식

컬러 키워드 : CSS 자체에서 사용 가능한 문자 식별자입니다. red, blue, black 등과 같이 미리 정의되어 있는 키워드를 이용해 색상을 표현할 수 있습니다. * 참고 : transparent는 투명성을 나타내는 키워드 *

16진법 ex. #RRGGBB : 6자리의 16진수(0-9, A-F)는 각각 두 자리씩 세 가지 색상을 나타냅니다. 첫 번

째 두 자리는 적색(RR), 가운데 두 자리는 녹색(GG), 마지막 두 자리는 청색(BB)을 의미합니다. 각 자리의 알파벳은 대소문자를 구분하지 않습니다.

16진법 ex. #RGB : 6자리의 16진수에서 각각의 두 자리가 같은 값을 가지면 3자리로 축약하여 사용할 수 있습니다. 예를 들어, #aa11cc 는 #a1c로 축약하여 사용할 수 있습니다.

RGB() : RGB 값은 rgb(R, G, B)의 형태로 각 변수 값(R 적색, G 녹색, B 청색)의 강도를 정의합니다. 0~255의 정수로 된 값을 지정할 수 있으며, 0 → 255는 검정 → 흰색으로 값의 변화를 나타냅니다.

RGBA() : RGBA 값은 기존 RGB에서 A값이 추가된 형태입니다. rgb(R, G, B, A)의 형태로 각 변수는(R 적색, G 녹색, B 청색, A 투명도)의 강도를 정의합니다. A 값은 0 ~ 1 사이의 값을 지정할 수 있으며, 0.5와 같이 소수점으로 표기합니다. 0 → 1은 투명 → 불투명으로 값의 변화를 나타냅니다. 예를 들어, rgba(0, 0, 0, 0)는 투명한 색상을 가지게 됩니다.

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>color</title>
```

```
</head>
```

```
<body>
```

```
<h1 style="color: red"> heading </h1>
```

```
<h1 style="color: #ff0000"> heading </h1>
```

```
<h1 style="color: #f00"> heading </h1>
```

```
<h1 style="color: rgb(255,0,0)"> heading </h1>
```

```
<h1 style="color: rgba(255,0,0, 0.5)"> heading </h1>
```

```
</body>
```

```
</html>
```

https://developer.mozilla.org/en-US/docs/Web/CSS/color_value

https://www.w3schools.com/css/css3_colors.asp

4) 속성-background

- background 관련 속성

background-color :

기본값 : transparent

배경의 색상을 지정하는 속성입니다. 앞선 색상 강의에서 배운 색상 값 적용 방식과 같습니다.

background-image :

기본값 : none

배경으로 사용할 이미지의 경로를 지정하는 속성입니다. url의 경로는 절대 경로, 상대 경로 모두 사용 가능합니다. 만약 background-color에 색상이 적용된 상태에서 background-image로 사용된 이미지에 투명한 부분이 있다면, 그 부분에 background-color 색상이 노출됩니다.

background-repeat :

기본값 : repeat

이미지의 반복 여부와 방향을 지정하는 속성입니다. 기본값이 repeat이기 때문에 따로 설정하지 않으면 x, y축으로 반복되어서 표시됩니다. background-repeat의 값으로 사용할 수 있는 것들은 다음과 같습니다.

<속성값>

repeat : x, y축으로 모두 반복합니다.

repeat-x : x축 방향으로만 반복합니다.
repeat-y : y축 방향으로만 반복합니다.
no-repeat : 이미지를 반복하지 않습니다.

background-position :

기본값 : 0% 0%

요소에서 배경 이미지의 위치를 지정하는 속성입니다. x축, y축으로부터의 위치를 지정할 수 있으며, 값의 선언 순서는 x축, y축으로부터의 간격입니다. 만일 한쪽만 지정된다면 나머지는 중앙값(center)으로 적용됩니다.

<속성값>

% : 기준으로부터 % 만큼 떨어진 지점과 이미지의 % 지점이 일치하는 곳에 위치시킵니다.

px : 기준으로부터 px 만큼 떨어진 지점과 이미지의 (0,0) 지점이 일치하는 곳에 위치시킵니다.

키워드 : top, left, right, bottom, center 키워드를 사용할 수 있습니다. 키워드는 선언 순서와 관계없이 top, bottom은 y축 기준으로 하며 left, right는 x축을 기준으로 합니다.

background-attachment :

기본값 : scroll

화면 스크롤에 따른 배경 이미지의 움직임 여부를 지정하는 속성입니다.

<속성값>

scroll : 배경 이미지는 요소 자체를 기준으로 고정되어 있으며 내용과 함께 스크롤 되지 않습니다.

local : 배경 이미지는 요소의 내용을 기준으로 고정되어 있으며 내용과 함께 스크롤 됩니다.

fixed : 배경 이미지는 뷰포트를 기준으로 고정되어 있으며 스크롤에 영향을 받지 않습니다.

- 뷰포트 : 사용자가 시각적으로 볼 수 있는 웹페이지 영역을 의미합니다. 컴퓨터나 휴대폰과 같은 장치에 Display 요소가 표현되는 영역을 말합니다.

background 축약

background: [-color] [-image] [-repeat] [-attachment] [-position];

<코드실습>

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8">

<title>background</title>

<style>

div {

height: 500px;

background-color: yellow;

background-image: url(https://www.w3schools.com/CSSref/img_tree.gif);

background-repeat: no-repeat;

background-position: center top;

/* 축약형 */

background: yellow url(https://www.w3schools.com/CSSref/img_tree.gif) no-repeat center

top;

}

</style>

</head>

<body>


```

    <div> css background 속성 실습 </div>
</body>
</html>
https://www.w3schools.com/css/css\_background.asp

```

5) 속성-boxmodel

문서를 배치할 때 브라우저의 렌더링 엔진은 표준 CSS 기본 박스 모델에 따라 각 요소를 사각형 상자로 나타냅니다. CSS를 이용해 이 상자의 크기, 위치 및 속성(색상, 배경, 테두리 크기 등)을 변경할 수 있습니다.

- boxmodel 구성



Content 영역 : 요소의 실제 내용을 포함하는 영역입니다. 따라서 크기는 내용의 너비 및 높이를 나타냅니다.

Border 영역 : content 영역을 감싸는 테두리 선을 border라고 합니다.

Padding 영역 : content 영역과 테두리 사이의 여백을 padding이라고 합니다. content 영역이 배경, 색 또는 이미지가 있을 때 패딩 영역까지 영향을 미칩니다. 이에 따라 padding을 content의 연장으로 볼 수도 있습니다.

Margin 영역 : border 바깥쪽의 영역을 margin이라고 합니다. border 영역을 다른 요소와 구별하기 위해 쓰이는 빈 영역입니다. 즉, 주변 요소와의 여백(간격)을 margin을 이용해 지정할 수 있습니다.

<코드실습>

```

<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>box model</title>
  <style>
    div {
      margin: 50px;
      padding: 50px;
      border: 10px solid #000;
    }
  </style>
</head>
<body>
  <div> 박스 모델에 대하여 알아보시다 </div>
</body>

```

</html>

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Box_Model/Introduction_to_the_CSS_box_model

6) 속성-border

- border 관련 속성

border-width

기본값 : medium

선의 굵기를 지정하는 속성입니다. border-top-width, border-bottom-width, border-right-width, border-left-width를 이용하여 상하좌우 선의 굵기를 다르게 표현할 수 있습니다.

border-width: [top] [right] [bottom] [left];

<속성값>

키워드 : thin, medium, thick

단위 : px, em, rem ... (% , 정수 단위 사용불가)

border-style

기본값 : none 선의 모양을 지정하는 속성입니다. border-top-style, border-bottom-style, border-right-style, border-left-style를 이용하여 상하좌우 선의 모양을 다르게 표현할 수 있습니다.

border-style: [top] [right] [bottom] [left];

또한, 위처럼 축약하여 공백을 이용해 각 방향에 대한 스타일을 지정할 수도 있습니다.

<속성값>

none : border를 표시하지 않습니다.

solid : border를 실선 모양으로 나타냅니다.

double : border를 이중 실선 모양으로 나타냅니다.

dotted : border를 점선 모양으로 나타냅니다.

그 밖에도 dashed, double, groove, ridge, inset, outset 등의 다양한 스타일이 있습니다.

border- color

기본값 : currentColor 선의 색상을 지정하는 속성입니다. border-top-color, border-bottom-color, border-right-color, border-left-color를 이용하여 상하좌우 선의 색상을 다르게 표현할 수 있습니다.

border-color: [top] [right] [bottom] [left];

또한, 위처럼 축약하여 공백을 이용해 각 방향의 색상을 지정할 수도 있습니다. 색상은 일반적인 CSS 색상 값 사용 방식과 같습니다.

border 축약

border: [-width] [-style] [-color];

위와 같이 공백으로 구분해 축약하여 사용할 수 있고, 정의되지 않은 속성값에 대해서는 기본값이 적용됩니다.

<코드실습>

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8">

<title>border</title>

<style>

div {

border-width: 10px;

```
border-style: solid;
border-color: #000;
```

```
/* 축약형 */
border: 10px solid #000;
}
</style>
</head>
<body>
    <div> css border 속성 실습 </div>
</body>
</html>
```

https://developer.mozilla.org/en-US/docs/Web/CSS/color_value#currentcolor_keyword
https://www.w3schools.com/css/css_border.asp

7) 속성-padding

- padding 속성

기본값 : 0

<속성값>

length : 고정값으로 지정합니다. (ex. px, em)

percent : 요소의 width에 상대적인 크기를 지정합니다.

padding-top : content 영역의 위쪽 여백을 지정합니다.

padding-right : content 영역의 오른쪽 여백을 지정합니다.

padding-bottom : content 영역의 아래쪽 여백을 지정합니다.

padding-left : content 영역의 왼쪽 여백을 지정합니다.

- syntax

padding: [-top] [-right] [-bottom] [-left];

0 10px 20px 30px /* 상, 우, 하, 좌 다름 */

0 10px 20px /* 좌, 우 같음 */

0 10px /* 상, 하 같음 & 좌, 우 같음 */

0 /* 상, 우, 하, 좌 모두 같음 */

기본적으로 padding의 4가지 속성을 위와 같이 축약하여 사용할 수 있습니다. 속성의 순서는 고정되어있으며, 위쪽을 기준으로 시계방향으로 돌아간다고 생각하면 쉽습니다. 축약형으로 사용할 때 반드시 4개의 속성에 대한 값을 모두 적어야 하는 것은 아닙니다. 속성값은 1~4개의 값을 사용할 수 있으며 border에서는 축약형 사용 시 정의되지 않은 속성값에 대해서 기본값이 적용되었지만, padding은 조금 다른 방식으로 동작합니다. 그 이유는 상하, 좌우 영역의 값이 같을 때 하나로 합쳐서 적용할 수도 있기 때문입니다.

padding : 20px 30px 40px 30px 일 때, 좌우의 패딩 값이 같을 때 padding : 20px 30px 40px 와 같이 함축하여 사용할 수 있습니다.

padding : 20px 30px 20px 일 때, 좌우 패딩과 마찬가지로 상하의 패딩 값이 같을 때 padding : 20px 30px 와 같이 함축하여 사용할 수 있습니다.

padding : 20px 20px (= 20px, 20px, 20px, 20px)일 때, 상하좌우 패딩 값이 모두 같을 때 padding : 20px 와 같이 하나의 값으로 함축하여 사용할 수 있습니다.

* 참고 : CSS에서 0 값에 대해서는 단위를 따로 적지 않습니다. 0px = 0% = 0em = 0pt... => " 0 "

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>padding</title>
```

```
  <style>
```

```
    div {
```

```
      padding-top: 10px;
```

```
      padding-right: 20px;
```

```
      padding-bottom: 40px;
```

```
      padding-left: 20px;
```

```
      /* 축약형 */
```

```
      padding: 10px 20px 40px;
```

```
    }
```

```
  </style>
```

```
</head>
```

```
<body>
```

```
  <div> css padding 속성 실습 </div>
```

```
</body>
```

```
</html>
```

<https://developer.mozilla.org/en-US/docs/Web/CSS/padding>

8) 속성-margin

- margin 속성

기본값 : 0

<속성값>

length : 고정값으로 지정합니다. (ex. px, em)

percent : 요소의 width에 상대적인 크기를 지정합니다.

auto : 브라우저에 의해 계산된 값이 적용됩니다.

margin-top : border 영역의 위쪽 여백을 지정합니다.

margin-right : border 영역의 오른쪽 여백을 지정합니다.

margin-bottom : border 영역의 아래쪽 여백을 지정합니다.

margin-left : border 영역의 왼쪽 여백을 지정합니다.

- syntax

margin: [-top] [-right] [-bottom] [-left];

0 10px 20px 30px /* 상, 우, 하, 좌 다름 */

0 10px 20px /* 좌, 우 같음 */

0 10px /* 상, 하 같음 & 좌, 우 같음 */

0 /* 상, 우, 하, 좌 모두 같음 */

padding과 마찬가지로 축약하여 사용할 수 있고, 상하, 좌우에 대해서 값이 같으면 하나로 합하여 사용

할 수 있습니다. margin에서는 수치 이외에 사용할 수 있는 'auto' 값이 있습니다.

margin auto : 기본적으로 브라우저에 의해 계산이 이루어지는데, 대부분의 경우 0(기본값) 또는 요소의 해당 측면에서 사용 가능한 공간과 같은 값을 가집니다. 이를 활용하여 수평 중앙 정렬을 할 수 있습니다. 아래 코드를 살펴봅시다.

```
margin-left: auto;
```

```
margin-right: auto;
```

좌우의 margin이 모두 auto로 적용되었다면, 브라우저는 요소가 가질 수 있는 가로 영역에서 자신의 width를 제외한 나머지 여백에 크기에 대해 균등 분할 하여 적용합니다. 이에 따라 요소는 수평 중앙 정렬이 됩니다. 상하의 경우 수직 중앙 정렬이 되지 않으며, 기본적인 플로우를 벗어나는 상황에 대해서 적용이 됩니다. 이는 좀 더 심화적인 개념이 필요하므로 기초에서는 수평 정렬에 대한 개념을 확실히 잡아두시길 바랍니다.

- margin collapse(마진 병합) : 마진 병합은 인접한 두 개 이상의 수직 방향 박스의 마진이 하나로 합쳐지는 것을 의미합니다.

1. 두 요소가 상하로 인접한 경우: 위 요소의 하단 마진과 아래 요소의 상단 마진의 병합이 일어납니다.

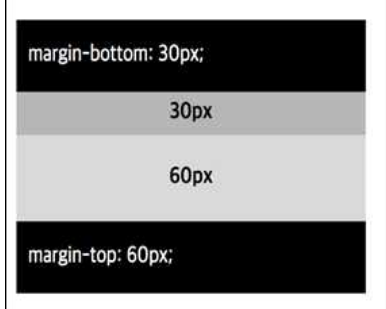
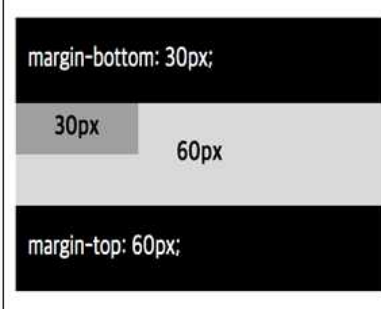
2. 부모 요소와 첫 번째 자식 요소 또는 마지막 자식 요소

부모 요소의 상단 마진과 첫 번째 자식 요소의 상단 마진 병합이 일어납니다.

부모 요소의 하단 마진과 마지막 자식 요소의 하단 마진 병합이 일어납니다.

3. 내용이 없는 빈 요소의 경우: 해당 요소의 상단 마진과 하단 마진의 병합이 일어납니다.

마진 병합은 수직 방향으로만 이루어지며, 좌우에 대해서는 일어나지 않습니다. 마진 병합은 마진이 반드시 맞닿아야 발생하기 때문에 2, 3번째의 경우 패딩 및 보더가 없어야 합니다.

마진 병합의 잘못된 예 (X)	마진 병합의 올바른 예 (O)
	

마진 병합을 활용하여 첫 번째와 두 번째 컴포넌트의 조합이 다양한 경우 여백을 다르게 사용 가능
<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>margin</title>
```

```
<style>
```

```
div {
```

```
margin-top: 10px;
```

```
margin-right: 20px;
```

```
margin-bottom: 10px;
```

```
margin-left: 20px;
```

```

        /* 축약형 */
        margin: 10px 20px ;
    }
</style>
</head>
<body>
    <div> css margin 속성 실습 </div>
</body>
</html>

```

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Box_Model/Mastering_margin_collapsing

https://www.w3schools.com/css/css_margin.asp

9) 속성-margin&padding

- margin과 padding의 비교

	+	-	auto	단위
margin	o	o	o	px, %...
padding	o	x	x	px, %...

음수값 사용 가능 여부 : 왜 margin은 음수 값 적용이 가능하고, padding은 적용되지 않을까요? 예를 들어 생각해보자면, padding은 뼈와 우리 피부 사이의 지방이라고 생각하고, margin은 사람과 사람 사이의 간격이라고 생각하면 쉽습니다. 지방은 아무리 뺀다고 해서 피부가 뼈보다 밑으로 갈 수 없을 뿐만 아니라, 0 이하가 될 수 없으므로 양수만 된다고 생각하면 됩니다. 그러나 사람과 사람 사이는 멀리 떨어질 수도 있지만, 서로 겹쳐서 서 있을 수도 있으므로 음수 값이 가능하다고 생각하면 됩니다.

%값의 사용과 기준점 : css 속성을 사용하면서 어떤 값을 적용할 때 이 단위를 적용 할 수 있을까? 라는 생각을 가지고 코딩하는 자세는 매우 중요합니다. margin과 padding은 px과 같은 고정적인 단위 외에도 %라는 상대적인 단위를 사용 할 수 있습니다. %는 요소의 크기를 기준으로 상대적인 값을 결정짓게 됩니다. 얼핏 생각하면, 상하는 height 값에 대해 좌우는 width 값에 대해 크기가 계산될 거 같지만 그렇지 않습니다. %는 상하좌우의 방향에 관계 없이 모두 요소의 width 값을 기준으로 값이 결정됩니다.

```

div {
    width: 100px;
    height: 200px;
    margin: 10%;
    padding: 10%;
}

```

만약 위와 같은 코드의 경우에는 margin과 padding이 모두 20px 10px 20px 10px으로 적용되는 것이 아니라, 10px 10px 10px 10px 값으로 적용됩니다.

```

<코드실습>
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>margin & padding</title>
    <style>
        div {

```

```

width: 300px;
height: 100px;
margin: -10px -10px 20px 30px;
padding: 10px 20px 30px 40px; /* padding 음수 사용 불가 */
}
</style>
</head>
<body>
  <div> css margin & padding 비교 실습 </div>
</body>
</html>

```

10) 속성 - width

- width 속성

기본값 : 0

<속성값>

auto : 브라우저에 의해 자동으로 계산하여 적용합니다. * 요소의 레벨 기본 특성에 따라 다르게 동작합니다.

length : 고정값으로 지정합니다. (ex. px, em)

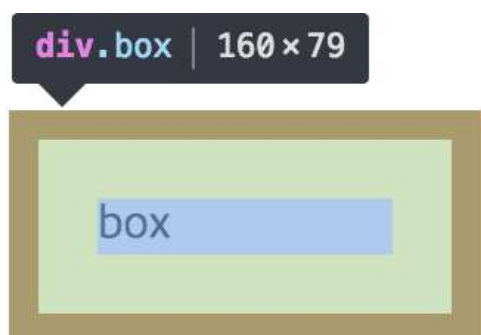
percent : 부모 요소의 width에 상대적인 크기를 지정합니다.

width는 content 영역의 너비를 제어할 때 사용하는데 이때 auto가 아닌 특정한 값을 지정하여 사용하면, 그 크기는 box model 영역에서 margin 영역을 제외한 모든 영역에 대해 영향을 받습니다. (content, padding, border) 예를 들어,

```
<div class="box"> box </div>
```

```
.box {
  width: 100px;
  padding: 20px;
  border: 10px solid black;
}
```

위와 같이 선언되어있다면 요소의 총 가로 크기는 160px가 됩니다. 분명 width: 100px를 적용했는데 width는 padding, border 영역에 대해서 영향을 받기 때문에 160px가 됩니다.



식으로 나타내면, $100\text{px content} + (20\text{px} * 2) \text{ padding} + (10\text{px} * 2) \text{ border} = 160\text{px}$ 가 된 것입니다. 또한, width는 %를 이용해서도 크기를 지정할 수 있습니다.

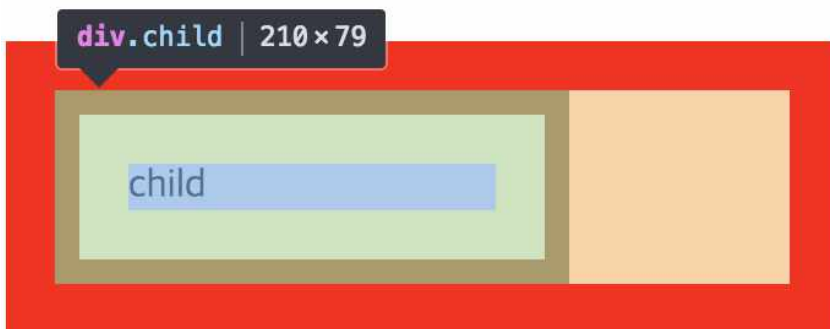
```
<div class="parent">
  <div class="child">
```

```

    child
  </div>
</div>

.parent {
  width: 300px;
  border: 20px solid red;
}
.child {
  width: 50%;
  padding: 20px;
  border: 10px solid black;
}

```



정답은 210px입니다. 우선, padding (20px * 2) + border (10px * 2) = 60px입니다. 210px - 60px = 150px 이며, 150px은 부모의 width가 300px이므로 300px의 50%인 150px이 width값으로 결정된 것입니다. 결국 %값일 때에는 부모의 width값에 대해서 계산되어지는데, 이때 부모의 width는 content 영역의 크기를 의미합니다. 부모의 padding과 border까지 더해진 요소의 전체 크기가 아닌, content 영역의 크기가 기준이라는 것을 기억하시면 됩니다.

<https://developer.mozilla.org/en-US/docs/Web/CSS/width>

11) 속성-height

- height 속성

기본값 : 0

<속성값>

auto : 브라우저 자동으로 계산하여 적용합니다. * 기본적으로 콘텐츠 영역의 내용만큼 높이를 가집니다.

length : 고정값으로 지정합니다. (ex. px, em)

percent : 부모 요소의 height에 상대적인 크기를 지정합니다. * 단, 부모 요소가 명시적으로 height 값이 있어야 합니다.

height는 content 영역의 높이를 제어할 때 사용하는데 이때 auto가 아닌 특정한 값을 지정하여 사용하면, width 속성과 마찬가지로 box model 영역에서 margin 영역을 제외한 모든 영역에 대해 영향을 받습니다. 예를 들어,

```
<div class="box"> box </div>
```

```

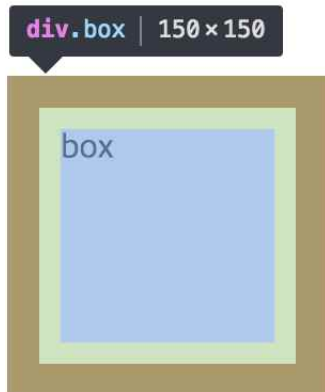
.box {
  width: 100px;
  height: 100px;
  padding: 10px;
}

```



```
border: 15px solid black;
}
```

위와 같이 선언되어있다면 요소의 총 세로 크기는 150px이 됩니다. 앞선 설명에서 언급한 바와 같이 height도 padding, border 영역에 대해서 추가로 영향을 받기 때문입니다.



식으로 나타내면, 100px content + (10px * 2) padding + (15px * 2) border = 150px 가 된 것입니다. %를 이용해서도 크기를 지정 할 수 있습니다.

```
<div class="parent">
  <div class="child">
    child
  </div>
</div>
```

```
.parent {
  width: 200px;
  border: 10px solid black;
}
.child {
  height: 50%;
  background: red;
}
```



실제로 확인해보면, height: auto일 때와 height: 50%일 때 차이가 없는 것을 확인할 수 있습니다. 왜 두 값의 차이가 없는 걸까요? MDN에서 height 속성에 percent value에 대한 설명을 보면 "Containing Block의 높이에 대한 백분율로 높이를 정의합니다."고 나와 있습니다. 여기서 말하는 Containing Block은 부모를 의미한다고 생각하시면 됩니다. 즉, 현재 위의 코드에서는 부모가 명시적인 높이 값을 가지고 있지 않기 때문에 자식의 높이를 % 값으로 지정해줘도 적용되지 않았던 것입니다.

부모 코드에 height: 200px로 명시적으로 높이 값을 지정해주면, 위와 같이 %값이 적용되는걸 볼 수 있습니다.

```
<코드실습>
<!DOCTYPE html>
<html lang="ko">
```



```
<head>
  <meta charset="UTF-8">
  <title>height</title>
  <style>
    div {
      height: 300px;
      margin: 10px 20px;
      padding: 30px;
      border: 20px solid black;
    }
  </style>
</head>
<body>
  <div> css height 속성 실습 </div>
</body>
</html>
```

<https://developer.mozilla.org/en-US/docs/Web/CSS/height>

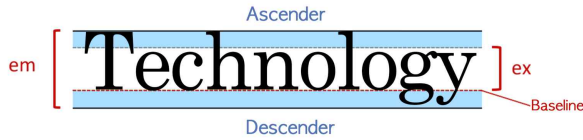
12) 속성-boxmodel정리

박스 모델은 content, padding, border, margin의 총 4가지 영역으로 나누어진다는 걸 배웠습니다. 추가적으로 content 영역의 너비는 width, 높이는 height를 통해서 제어 할 수 있으며, width와 height 그리고 padding과 border를 모두 더한 것이 요소의 전체 크기가 된다는 것에 대해 배웠습니다. 다른 요소와의 여백은 margin을 이용하여 줄 수 있으며 음수값을 사용할 수 있습니다. 또한, margin은 상하 요소 사이의 병합 현상이 일어나고 이때는 큰 값을 기준으로 병합된다는 걸 기억!!

6. 폰트, 텍스트

1) 속성-typography

- 타이포그래피의 구조 : 모든 폰트는 em박스를 가지고 있고 위 그림과 같은 구조로 이루어져 있습니다.



em : 폰트의 전체 높이를 의미

ex(= x-height) : 해당 폰트의 영문 소문자 x의 높이를 의미

Baseline : 소문자 x를 기준으로 하단의 라인을 의미

Descender : 소문자에서 baseline 아래로 처지는 영역을 의미, 서체에 따라 descender의 길이가 다릅니다. (g, j, p, q, y)

Ascender : 소문자 x의 상단 라인 위로 넘어가는 영역을 의미 (b, d, h, l)

<https://cssreference.io/typography/>

2) 속성-font-family

- font-family 속성 : 글꼴을 지정하는 속성

font-family : family-name | generic-family (| initial | inherit);

family-name: 사용할 폰트의 이름을 나타내며 ' , ' 로 구분하여 여러 개 선언할 수 있습니다. 먼저 선언된 순서대로 우선순위가 결정됩니다. 이름 중간에 공백이 있거나, 한글일 경우 홑따옴표로 묶어서 선언합니다.

generic-family : family-name으로 지정된 글꼴을 사용할 수 없을 경우를 대비해, 브라우저가 대체할 수 있는 폰트가 필요한 경우 선택할 수 있게 해줍니다. font-family 속성의 맨 마지막에 선언해야 하며, 키워드이기 때문에 따옴표 등의 인용부호로 묶지 않는 것이 원칙입니다. 예를 들면 아래와 같이 선언하여 사용할 수 있습니다.

font-family: Helvetica, Dotum, '돋움', Apple SD Gothic Neo, sans-serif;

가장 먼저 Helvetica를 사용하고, 이를 사용할 수 없을 때 Dotum을 사용하는 방식으로 우선순위에 따라 차례대로 적용 됩니다. 만약 "abc 가나다 123"이라는 글자가 있다면, "abc"와 "123"은 Helvetica로 표현이 되고, "가나다"는 Dotum으로 표현이 됩니다. "가나다"가 Dotum으로 표현된 이유는 Helvetica는 한글을 지원하는 폰트가 아니기 때문입니다. 그리고 예를 보면 돋움체를 영문으로 한번, 한글로 한번 선언 하였습니다. 한글을 지원하지 않는 디바이스일 경우 해당 한글 폰트를 불러올 수 없으므로 영문명으로도 선언해 주어야 합니다. 마지막에는 반드시 generic-family를 선언해 주어야 합니다. 그 이유는 선언된 모든 서체를 사용할 수 있다는 보장이 없기 때문입니다. 이때 generic-family를 선언해주면, 시스템 폰트 내에서 사용자가 의도한 스타일과 유사한 서체로 적용되기 때문입니다. 또한, 자식 요소에서 font-family를 재선언하면 부모에 generic-family가 선언되어있어도 다시 선언해주어야 합니다. Generic-Family에는 대표적인 서체로 serif, sans-serif가 있습니다. serif는 삐침이라는 뜻이고, sans는 프랑스어로 '~이 없이'라는 의미가 있습니다. serif는 글자 획에 삐침이 있는 폰트로 대표적으로 명조체가 있으며, sans-serif는 획에 삐침이 없는 폰트로 대표적으로 돋움체가 있습니다.

<https://developer.mozilla.org/ko/docs/Web/CSS/font-family>

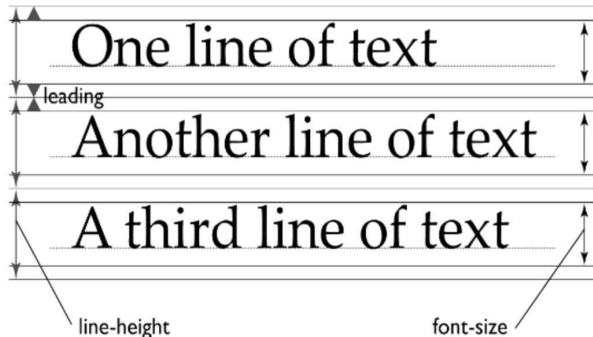
<https://qwerty.dev/>

3) 속성-line-height

- line-height 속성 : line-height는 텍스트 라인의 높이를 의미하는 것으로 주로 행간을 제어할 때 사용됩니다.

기본값 : normal

line-height: normal | number | length | initial | inherit ;



행간을 제어할 때 사용하는 속성이라 해서 줄 간격으로 생각해 오해하기 쉬울 수가 있습니다. 줄 바꿈이 되었을 때, 윗줄의 텍스트 하단과 아랫줄의 텍스트 상단까지의 간격이라고 생각할 수도 있지만, line-height로 제어되는 부분을 line-box라고도 하며 이는 타이포그래피 구조에서 배웠던 [em 박스] + [상하단의 여백]까지를 의미합니다.

<속성값>

normal : 기본값으로 브라우저의 기본 속성을 따릅니다. 폰트에 따라 브라우저에 따라 다르지만 보통 1.2 정도로 할당되어 있습니다.

number : font-size를 기준으로 설정한 숫자만큼 배율로 적용합니다.

length : px, em 등 고정 수치로 할당할 수 있습니다.

% : font-size를 기준으로 설정한 퍼센트만큼 배율로 적용합니다.

주의할 점은, line-height의 값으로 number를 선언할 때와 %로 선언할 때의 차이입니다. 두 값 모두 font-size를 기준으로 동작하기 때문에 1이나 100%를 같은 것이라고 오해할 수 있습니다. 하지만 두 값은 큰 차이가 있습니다. 바로 line-height의 값이 자식 요소로 상속되었을 때의 계산 방식입니다.

number : 부모 요소의 숫자 값이 그대로 상속됩니다. 즉, 자식 요소에서도 또 한 번 자식 요소의 font-size를 기준으로 계산된 값을 가집니다.

% : 부모 요소에서 %값이 그대로 상속되는 것이 아니고, %에 의해 이미 계산된 px값이 상속됩니다.

```
body { font-size: 20px; line-height: 2; } /* line-height = 40px; */
```

```
body { font-size: 20px; line-height: 200%; } /* line-height = 40px; */
```

두 경우 모두 <body>에 똑같이 line-height: 40px이 적용됩니다. 하지만 자식 요소로 <p>가 있다고 생각을 하면 얘기가 달라집니다.

```
body { font-size: 20px; line-height: 2; } /* line-height = 40px; */
```

```
p { font-size: 10px; } /* line-height = 20px; */
```

```
body { font-size: 20px; line-height: 200%; } /* line-height = 40px; */
```

```
p { font-size: 10px; } /* line-height = 40px; */
```

계산된 값이 아닌 숫자 값을 상속한다는 사실 때문에, 숫자 값을 사용하면 부모 엘리먼트에서 계산된 값 대신 비율을 그대로 상속받을 수 있으므로, 가능하면 단위가 없는 값을 사용하는 것이 좋습니다.

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```

<meta charset="UTF-8">
<title>line-height</title>
<style>
    .parent {
        width: 200px;
        font-size: 10px;

        line-height: normal;
        line-height: 20px;
        line-height: 2;
        line-height: 200%;
    }

    .child {
        font-size: 20px;
    }
</style>
</head>

<body>
    <div class="parent">
        <div class="child">
            Lorem ipsum dolor sit amet consectetur adipisicing elit.
            Ipsam aspernatur vitae sapiente laudantium velit quo unde cupiditate autem,
            harum eaque natus perferendis ducimus saepe libero, voluptatibus voluptates possimus.
            Adipisci, delectus.
        </div>
    </div>
</body>
</html>

```

https://www.w3schools.com/cssref/pr_dim_line-height.asp

<https://wit.nts-corp.com/2017/09/25/4903>

<https://developer.mozilla.org/en-US/docs/Web/CSS/line-height>

4) 속성-font-size

- font-size 속성 : 글꼴의 크기를 지정하는 속성

기본값 : medium

font-size: keyword | length | initial | inherit ;

<속성값>

keyword : medium(기본 값), xx-small, x-small, small, large, x-large, xx-large, smaller, larger

length : px, em 등 고정 수치로 지정합니다.

% : 부모 요소의 font-size 기준의 퍼센트로 지정합니다.

absolute size (keyword) : 기본 값인 medium에 대한 상대적인 크기로, 브라우저마다 사이즈가 다르게 정의되어있습니다.

relative size (keyword) : 부모 요소의 font-size 크기에 대해 상대적입니다. smaller는 0.8배, larger는 1.2배입니다.

length : px, em, rem 등의 단위를 이용하여 고정된 크기를 지정할 수 있습니다. - em : 부모 요소의 font-size에 em 값을 곱한 크기 - rem : 루트의 font-size에 rem 값을 곱한 크기

percent (%) : 부모 요소의 font-size를 기준으로 백분을 계산된 값을 지정할 수 있습니다.

viewport units : vw, vh 단위로 뷰포트를 기준으로 하여, 유동적인 font-size를 지정할 수 있습니다. vw는 뷰포트 width의 1%, vh는 뷰포트 height의 1% 값을 가집니다.

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>font-size</title>
```

```
<style>
```

```
.parent {
```

```
    font-size: 20px;
```

```
}
```

```
.child {
```

```
    font-size: 1em;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class="parent">
```

```
<div class="child">
```

```
    Lorem ipsum dolor sit amet consectetur adipisicing elit.
```

```
    Ipsam aspernatur vitae sapiente laudantium velit quo unde cupiditate autem,
```

```
    harum eaque natus perferendis ducimus saepe libero, voluptatibus voluptates possimus.
```

```
    Adipisci, delectus.
```

```
</div>
```

```
</div>
```

```
</body>
```

```
</html>
```

https://www.w3schools.com/cssref/pr_font_font-size.asp

<https://developer.mozilla.org/ko/docs/Web/CSS/font-size>

5) 속성-font-weight

- font-weight 속성 : 글꼴의 굵기를 지정하는 속성

기본값 : normal

font-weight: normal | bold | bolder | lighter | number | initial | inherit ;

<속성값>

normal : 기본값 (400)

bold : 굵게 표현(700)

bolder : 부모 요소 보다 두껍게 표현

lighter : 부모 요소 보다 얇게 표현

number : 100, 200, 300, 400, 500, 600, 700, 800, 900 (클수록 더 두껍게 표현)

* 실무에서는 normal과 bold를 많이 사용하고, 부모 요소에 영향이 있는 bolder와 lighter는 사용될 수 있으면 지양하는 편입니다. 물론 상속 관계에서 바뀌어야 하는 스펙이라면 당연히 유용하게 사용될 수 있지만, 그 외의 경우에는 사용에 있어 신중해야 합니다. font-weight는 normal, bold와 같은 키워드 외에 숫자로도 그 굵기를 표현할 수 있습니다. 100~900까지 100단위로 값을 지정하여 사용할 수 있고 숫자가 커질수록 더욱 굵게 표현됩니다. 기본적으로 400은 normal과 같고, 700은 bold와 같습니다. 그러나 수치를 이용한 font-weight는 폰트 자체에서 지원을 해야 표현할 수 있습니다. 폰트에 따라 font-weight를 적용해도 굵기에 변화가 없을 수도 있으며, normal과 bold만 지원하는 폰트일 경우에는 100~500까지는 normal로, 600~900까지는 bold로 표현됩니다. 폰트가 점차 다양해지면서 굵기 자체를 폰트 family-name으로 가지고 있는 경우도 있습니다. 또한, 디바이스별로 조금 다르게 표현될 수도 있습니다. font-weight와 font-family, font-size는 서로 밀접하게 연관되어있습니다.

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>font-size</title>
```

```
  <link href="//fonts.googleapis.com/css?family=Open+Sans:400,600,700,800,300" rel="stylesheet" type="text/css">
```

```
  <style>
```

```
    body {
```

```
      padding: 0 20px;
```

```
      font-family: 'Open Sans';
```

```
    }
```

```
    .w100 { font-weight: 100; }
```

```
    .w200 { font-weight: 200; }
```

```
    .w300 { font-weight: 300; }
```

```
    .w400 { font-weight: 400; }
```

```
    .w500 { font-weight: 500; }
```

```
    .w600 { font-weight: 600; }
```

```
    .w700 { font-weight: 700; }
```

```
    .w800 { font-weight: 800; }
```

```
    .w900 { font-weight: 900; }
```

```
  </style>
```

```
</head>
```

```
<body>
```

```
  <p class="w100">This is 100 weight</p>
```

```
  <p class="w200">This is 200 weight</p>
```

```
  <p class="w300">This is 300 weight (available)</p>
```

```
  <p class="w400">This is 400 weight (available)</p>
```

```
  <p class="w500">This is 500 weight</p>
```

```
  <p class="w600">This is 600 weight (available)</p>
```

```
  <p class="w700">This is 700 weight (available)</p>
```

```
<p class="w800">This is 800 weight (available)</p>
<p class="w900">This is 900 weight</p>
</body>
</html>
https://www.w3schools.com/cssref/pr\_font\_weight.asp
https://codepen.io/impressivewebs/pen/EIncg
https://developer.mozilla.org/ko/docs/Web/CSS/font-weight
```

6) 속성-font-style

- font-style 속성 : 글꼴의 스타일을 지정하는 속성
기본값 : normal
font-style: normal | italic | oblique | initial | inherit;

This should be the Regular face

This should be the Italic face

This should be the Oblique face

<속성값>
normal : font-family 내에 분류된 기본값
italic : italic 스타일로 표현합니다.
oblique : oblique 스타일로 표현합니다.
oblique 텍스트의 기울기에 대한 각도를 추가로 지정할 수 있습니다.
font-weight oblique <각도>;
유효한 값은 -90 ~ 90도이며, 따로 각도를 지정하지 않으면 14도가 사용됩니다. 양수 값은 글의 끝부분 쪽으로 기울어지며, 음수값은 시작 부분 쪽으로 기울어집니다. 그러나 아직 초안 단계로 CSS Fonts Module Level 4를 지원하는 브라우저에서만 사용 가능합니다.
<https://developer.mozilla.org/ko/docs/Web/CSS/font-style>
https://www.w3schools.com/cssref/pr_font_font-style.asp

7) 속성-font-variant

- font-variant 속성 : 글꼴의 형태를 변형하는 속성으로 소문자를 작은 대문자로 변환할 수 있습니다.
기본값 : normal
font-variant: normal | small-caps | initial | inherit ;
<속성값>
normal : 기본값
small-caps : 소문자를 작은 대문자로 변형합니다.
<코드실습>
<!DOCTYPE html>
<html lang="ko">
<head>
<meta charset="UTF-8">
<title>font-variant</title>
<style>


```

    p {
        font-variant: small-caps;
    }
</style>
</head>

```

```

<body>
    <p>Font-Variant: Small-Caps</p>
</body>
</html>

```

https://www.w3schools.com/cssref/pr_font_font-variant.asp
<https://developer.mozilla.org/en-US/docs/Web/CSS/font-variant>

8) 속성-font

- font 관련 속성 : font-style, font-variant, font-weight, font-size/line-height, font-family 속성들을 한 번에 선언할 수 있는 축약형 속성입니다.

기본값 : 각 속성들의 기본값

font: font-style font-variant font-weight font-size/line-height font-family | initial | inherit;

<속성값>

font-style : font-style 지정, 기본 값은 normal

font-variant : font-variant 지정, 기본 값은 normal

font-weight : font-weight 지정, 기본 값은 normal

font-size/line-height : font-size/line-height 지정, 기본 값은 normal

font-family : font-family 지정

/* style | size | family */

font: oblique 2em "돋움", dotum, sans-serif;

/* style | variant | weight | size/line-height | family */

font: oblique small-caps bold 16px/1.5 '돋움';

/* The font used in system dialogs */

font: message-box;

font: icon;

축약형을 선언할 때 유의사항

font-size와 font-family는 반드시 선언해야 하는 필수 속성입니다.

빠진 속성이 있다면 기본값으로 지정됩니다.

각 속성의 선언 순서를 지켜야 합니다.

https://www.w3schools.com/cssref/pr_font_font.asp

<https://developer.mozilla.org/en-US/docs/Web/CSS/font>

9) 속성-webfont

- @font-face : 웹에 있는 글꼴을 사용자의 로컬 환경(컴퓨터)으로 다운로드하여 적용하는 속성입니다.

기본값 : 없음

@font-face {

```

font-properties
}
<속성값>
font-family(필수) : 글꼴의 이름을 지정
src(필수) : 다운로드 받을 글꼴의 경로(URL)

font-style(옵션) : 글꼴의 스타일 지정, 기본값은 normal
font-weight(옵션) : 글꼴의 굵기 지정, 기본값은 normal
@font-face {
    font-family: webNanumGothic; /* 사용자 지정 웹 폰트명 */
    src: url(NanumGothic.eot); /* 적용 될 웹 폰트의 경로 */
    font-weight: bold; /* 필요에 따라 지정 */
    font-style: italic; /* 필요에 따라 지정 */
}

```

```

body {
    font-family: webNanumGothic;
}

```

웹 폰트는 선언 시 필요에 따라 굵기나 스타일 등을 같이 지정해서 사용할 수 있습니다.

<https://wit.nts-corp.com/2017/02/13/4258>

<https://web.dev/fast/#optimize-webfonts>

10) 속성-vertical-align

- vertical-align 속성 : 요소의 수직 정렬을 지정하는 속성

기본값 : baseline

vertical-align: keyword | length | percent | initial | inherit ;

<속성값>

length : 요소를 지정한 길이만큼 올리거나 내림. 음수 허용

% : 요소를 line-height를 기준으로 올리거나 내림. 음수 허용

keyword : baseline(기본값), sub, super, top, text-top, middle, bottom, text-bottom

vertical-align은 기본값이 baseline입니다. 이전에 타이포그래피 구조 강의에서 설명했듯이 baseline은 소문자 x를 기준으로 하단 라인을 의미합니다.

keyword sub : 부모 아래 첨자 기준으로 정렬 super : 부모 위 첨자 기준으로 정렬 text-top : 부모 텍스트의 맨 위(Ascender 제외) text-bottom : 부모의 텍스트의 맨 아래(Descender 제외) middle : 부모의 중앙에 위치 top : 부모의 맨 위 위치 bottom : 부모의 맨 아래 위치

length : px값 사용 시 baseline을 기준으로 이동하며, 음수 값도 사용 가능합니다.

percent(%) : line-height를 기준으로 내에서 이동하며 음수 값 사용 가능합니다.

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>vertical-align</title>
```

```
<style>
```

```

p {
  padding: 10px;
  border: 1px dashed #aaa;
  line-height: 1;
  font-size: 16px;
}

p span {
  background-color: rgba(0, 255, 255, 0.5);
  border: 1px solid #aaa;
}

p span:nth-child(1) {
  background-color: rgba(255, 255, 0, 0.5);
}

p span:nth-child(2),
p span:nth-child(4) {
  font-weight: bold;
  font-size: 20px;
}

p span:nth-child(3) {
  background-color: rgba(0, 0, 0, 0.2);
}

/* table */

table {
  width: 100%;
  border-collapse: collapse;
}

table td,
table th {
  border: 1px solid #aaa;
  height: 50px;
}
</style>
</head>
<body>
  <h1>vertical-align</h1>
  <p><span>vertical-align:</span>
    <span style="vertical-align: baseline;">baseline;</span>

```

```

    <span>---</span>
    <span style="vertical-align: baseline;">수직정렬</span></p>
<p><span>vertical-align:</span>
    <span style="vertical-align: sub;">sub;</span>
    <span>---</span>
    <span style="vertical-align: sub;">수직정렬</span></p>
<p><span>vertical-align:</span>
    <span style="vertical-align: super;">super;</span>
    <span>---</span>
    <span style="vertical-align: super;">수직정렬</span></p>
<p>
    <span>vertical-align:</span>
    <span style="vertical-align: text-top;">text-top;</span>
    <span>---</span>
    <span style="vertical-align: text-top;">수직정렬</span></p>
<p><span>vertical-align:</span>
    <span style="vertical-align: text-bottom;">text-bottom;</span>
    <span>---</span>
    <span style="vertical-align: text-bottom;">수직정렬</span></p>
<p><span>vertical-align:</span>
    <span style="vertical-align: middle;">middle;</span>
    <span>---</span>
    <span style="vertical-align: middle;">수직정렬</span></p>
<p><span>vertical-align:</span>
    <span style="vertical-align: top;">top;</span>
    <span>---</span>
    <span style="vertical-align: top;">수직정렬</span></p>
<p><span>vertical-align:</span>
    <span style="vertical-align: bottom;">bottom;</span>
    <span>---</span>
    <span style="vertical-align: bottom;">수직정렬</span></p>
<p><span>vertical-align:</span>
    <span style="vertical-align: 4em;">4em;</span>
    <span>---</span></p>
<p><span>vertical-align:</span>
    <span style="vertical-align: 4px;">4px;</span>
    <span>---</span></p>
<p><span>vertical-align:</span>
    <span style="vertical-align: 20%;">20%;</span>
    <span>---</span></p>
<p><span>vertical-align:</span>
    <span style="vertical-align: -10px;">-10px;</span>
    <span>---</span></p>

```

```

<table>
  <caption>table cell vertical-align</caption>
  <thead>
    <tr>
      <th>속성값</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td style="vertical-align:baseline;">vertical-align:baseline</td>
    </tr>
    <tr>
      <td style="vertical-align:top;">vertical-align:top</td>
    </tr>
    <tr>
      <td style="vertical-align:middle;">vertical-align:middle(기본값)</td>
    </tr>
    <tr>
      <td style="vertical-align:bottom;">vertical-align:bottom</td>
    </tr>
    <tr>
      <td>null</td>
    </tr>
  </tbody>
</table>
</body>
</html>

```

https://www.w3schools.com/cssref/pr_pos_vertical-align.asp
<https://developer.mozilla.org/ko/docs/Web/CSS/vertical-align>

11) 속성-text-align

- text-align 속성 : 텍스트의 정렬을 지정하는 속성입니다.

기본값 : left (Right to Left 언어일 경우는 right)

text-align: left | right | center | justify | initial | inherit ;

기본 값은 left이지만 경우에 따라 다릅니다. 문서의 방향이 LTR(Left To Right) 왼쪽에서 오른쪽 방향인 언어일 경우 left가 기본값이고, RTL(Right To Left) 로 오른쪽에서 왼쪽으로 읽힐 경우 right가 기본값이 됩니다.

<속성값>

left : 텍스트를 왼쪽으로 정렬

right : 텍스트를 오른쪽으로 정렬

center : 텍스트를 중앙으로 정렬

justify : 텍스트를 라인 양쪽 끝으로 붙여서 정렬. (마지막 라인은 정렬 하지 않음)

- text-align과 display의 관계 : text-align은 inline-level에 적용, text-align은 block-level에 적용할 수 없음

block 요소를 가운데 정렬 : 박스모델 챕터에서 다룬 margin의 auto
가운데 정렬 인라인 요소 : text-align (center) 블록 요소 : margin (auto)

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>text-align</title>
```

```
  <style>
```

```
    p {
```

```
      max-width: 630px;
```

```
      border: 1px solid #888;
```

```
      padding: 10px;
```

```
    }
```

```
  </style>
```

```
</head>
```

```
<body>
```

```
  <h1>text-align</h1>
```

```
  <h2>left</h2>
```

<p style="text-align: left;">이 안내서를 시작하기 전에, 사용법에 익숙한 텍스트 파일을 편집할 수 있는 편집 툴이 필요하며 최신 브라우저도 필요하다. 파일을 편집하고 싶지 않다면 그냥 안내서를 읽고 해당 예시 그림을 보라. 그러나 배우는데는 더딜 것이다.</p>

```
  <h2>right</h2>
```

<p style="text-align: right;">이 안내서를 시작하기 전에, 사용법에 익숙한 텍스트 파일을 편집할 수 있는 편집 툴이 필요하며 최신 브라우저도 필요하다. 파일을 편집하고 싶지 않다면 그냥 안내서를 읽고 해당 예시 그림을 보라. 그러나 배우는데는 더딜 것이다.</p>

```
  <h2>center</h2>
```

<p style="text-align: center;">이 안내서를 시작하기 전에, 사용법에 익숙한 텍스트 파일을 편집할 수 있는 편집 툴이 필요하며 최신 브라우저도 필요하다. 파일을 편집하고 싶지 않다면 그냥 안내서를 읽고 해당 예시 그림을 보라. 그러나 배우는데는 더딜 것이다.</p>

```
  <h2>justify</h2>
```

<p style="text-align: justify;">이 안내서를 시작하기 전에, 사용법에 익숙한 텍스트 파일을 편집할 수 있는 편집 툴이 필요하며 최신 브라우저도 필요하다. 파일을 편집하고 싶지 않다면 그냥 안내서를 읽고 해당 예시 그림을 보라. 그러나 배우는데는 더딜 것이다.</p>

```
</body>
```

```
</html>
```

https://www.w3schools.com/cssref/pr_text_text-align.asp

<https://developer.mozilla.org/en-US/docs/Web/CSS/text-align>

12) 속성-text-indent

- text-indent 속성 : 텍스트의 들여쓰기를 지정하는 속성입니다.

기본값 : 0

text-indent: length | initial | inherit;

<속성값>

length : px, em 등 고정 수치로 지정. 음수 허용

% : 부모 요소의 width를 기준으로 퍼센트로 지정

length : 문단의 첫 줄에 대한 들여쓰기를 수행합니다. 음수 값을 사용할 수 있으며, 음수 값 사용 시 왼쪽으로 이동합니다.

percent(%) : 텍스트를 포함하는 컨테이너 블록의 width(부모의 width)를 기준으로 변환된 백분율 값으로 들여쓰기합니다.

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>text-indent</title>
```

```
</head>
```

```
<body>
```

```
  <h1>text-indent</h1>
```

```
  <h2>length 1em</h2>
```

<p style="text-indent: 1em;">이 안내서를 시작하기 전에, 사용법에 익숙한 텍스트 파일을 편집할 수 있는 편집 툴이 필요하며 최신 브라우저도 필요하다. 파일을 편집하고 싶지 않다면 그냥 안내서를 읽고 해당 예시 그림을 보라. 그러나 배우는 데는 더딜 것이다.</p>

```
  <h2>length 40px</h2>
```

<p style="text-indent: 40px;">이 안내서를 시작하기 전에, 사용법에 익숙한 텍스트 파일을 편집할 수 있는 편집 툴이 필요하며 최신 브라우저도 필요하다. 파일을 편집하고 싶지 않다면 그냥 안내서를 읽고 해당 예시 그림을 보라. 그러나 배우는 데는 더딜 것이다.</p>

```
  <h2>percentage 15%</h2>
```

<p style="text-indent: 15%;">이 안내서를 시작하기 전에, 사용법에 익숙한 텍스트 파일을 편집할 수 있는 편집 툴이 필요하며 최신 브라우저도 필요하다. 파일을 편집하고 싶지 않다면 그냥 안내서를 읽고 해당 예시 그림을 보라. 그러나 배우는 데는 더딜 것이다.</p>

```
  <h2>length -40px (비추천)</h2>
```

<p style="text-indent: -40px;">이 안내서를 시작하기 전에, 사용법에 익숙한 텍스트 파일을 편집할 수 있는 편집 툴이 필요하며 최신 브라우저도 필요하다. 파일을 편집하고 싶지 않다면 그냥 안내서를 읽고 해당 예시 그림을 보라. 그러나 배우는 데는 더딜 것이다.
들여쓰기에 음수사용은 올바르지 않으며 본래의 기능에서 벗어남</p>

```
</body>
```

```
</html>
```

https://www.w3schools.com/cssref/pr_text_text-indent.asp

<https://developer.mozilla.org/en-US/docs/Web/CSS/text-indent>

<http://www.w3big.com/ko/cssref/pr-text-text-indent.html>

13) 속성-text-decoration

- text-decoration 속성 : 텍스트의 장식을 지정하는 속성입니다.

기본값 : none currentColor solid

text-decoration: text-decoration-line text-decoration-color text-decoration-style | initial | inherit;

text-decoration-line 텍스트 꾸밈의 종류를 지정하는 속성입니다.

기본값 : none

<속성값>

none : 텍스트 꾸밈을 생성하지 않음 (기본값)

underline : 밑줄로 꾸밈을 설정

overline : 윗줄로 꾸밈을 설정

line-through : 중간을 지나는 줄로 꾸밈을 설정

text-decoration-color

텍스트 꾸밈의 색상을 지정하는 속성입니다.

기본값 : currentColor

색상 값을 사용하여 원하는 색상을 지정할 수 있습니다.

text-decoration-style 꾸밈에 사용되는 선의 스타일을 지정하는 속성입니다.

기본값 : solid

<속성값>

solid : 한줄 스타일 (기본 값)

double : 이중선 스타일

dotted : 점선 스타일

dashed : 파선 스타일

wavy : 물결 스타일

<코드실습>

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8">

<title>text-decoration</title>

</head>

<body>

<h2>일반 경우</h2>

<p style="text-decoration: overline;">

text-decoration: overline;

</p>

<p style="text-decoration: underline;">

text-decoration: underline;

</p>

<p style="text-decoration: line-through;">

text-decoration: line-through;

</p>

<h2>부모 내 자식요소가 float될 경우 상속이 해제됨</h2>

text-decoration: overline;

<h2>부모 내 자식요소가 absolute 경우 상속이 해제됨</h2>


```

text-decoration: overline;</span>
</a>
</body>
</html>
https://www.w3schools.com/cssref/pr\_text\_text-decoration.asp
https://www.w3schools.com/cssref/css3\_pr\_text-decoration-line.asp
https://www.w3schools.com/cssref/css3\_pr\_text-decoration-color.asp
https://www.w3schools.com/cssref/css3\_pr\_text-decoration-style.asp
https://developer.mozilla.org/en-US/docs/Web/CSS/text-decoration

```

14) 속성-단어 관련 속성

- white-space 속성 : 요소 안에 공백을 어떻게 처리할지 지정하는 속성입니다.

기본값 : normal

white-space: normal | nowrap | pre | pre-line | pre-wrap | initial | inherit;

<속성값>

normal : 공백과 개행을 무시하고, 필요한 경우에 자동 줄 바꿈 발생. 기본값

nowrap : 공백과 개행을 무시하고, 자동 줄 바꿈이 일어나지 않음.

pre : 공백과 개행을 표현하고, 자동 줄 바꿈이 일어나지 않음.

pre-line : 공백은 무시하고, 개행만 표현. 필요한 경우에 자동 줄 바꿈 발생.

pre-wrap : 개행은 무시하고, 공백만 표현. 필요한 경우 자동 줄 바꿈 발생.

- letter-spacing 속성 : 자간을 지정하는 속성입니다.

기본값 : normal

letter-spacing: normal | length | initial | inherit;

<속성값>

normal : 기본값

length : 길이만큼 자간을 지정. 음수 허용

- word-spacing 속성 : 단어 사이의 간격을 지정하는 속성입니다.

기본값 : normal

word-spacing: normal|length|initial|inherit;

<속성값>

normal : 기본값

length : 길이만큼 단어 사이의 간격을 지정. 음수 허용

- word-break 속성 : 단어가 라인 끝에 나올 경우 어떻게 처리할지(중단점) 지정하는 속성입니다.

기본값 : normal

word-break: normal | break-all | keep-all | initial | inherit;

<속성값>

normal : 기본 값. 중단점은 공백이나 하이픈(-)(CJK는 음절)

break-all : 중단점은 음절. 모든 글자가 요소를 벗어나지 않고 개행

keep-all : 중단점은 공백이나 하이픈(-)(CJK는 그 외 기호도 포함)

- word-wrap 속성 : 요소를 벗어난 단어의 줄 바꿈을 지정하는 속성입니다.

기본값 : normal

word-wrap: normal|break-word|initial|inherit;

<속성값>

normal : 기본값. 중단점에서 개행

break-word : 모든 글자가 요소를 벗어나지 않고 강제로 개행

https://www.w3schools.com/cssref/pr_text_white-space.asp

https://www.w3schools.com/cssref/pr_text_letter-spacing.asp

https://www.w3schools.com/cssref/pr_text_word-spacing.asp

https://www.w3schools.com/cssref/css3_pr_word-break.asp

https://www.w3schools.com/cssref/css3_pr_word-wrap.asp

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Text

7. 레이아웃

1) 속성 - display

- display 속성 : 요소의 렌더링 박스 유형을 결정하는 속성입니다.

기본값 : -(요소마다 다름)

display: value;

<속성값>

none : 요소가 렌더링 되지 않음

inline : inline level 요소처럼 렌더링

block : block level 요소처럼 렌더링

inline-block : inline level 요소처럼 렌더링(배치)되지만 block level의 성질을 가짐

* height 나 width 등과 같은 박스모델 속성을 적용할 수 있다

그외에 list-item, flex, inline-flex, table, table-cell 등 다양한 속성값 존재

inline level 요소 사이의 공백과 개행 처리 inline 요소의 경우 공백과 개행에 대해서 하나의 여백으로 받아들입니다. 따라서 inline와 inline-block의 경우 태그 사이의 공백이나 개행이 있을 경우 약 4px의 여백을 가지게 됩니다.

- display와 box model의 관계

display	width	height	margin	padding	border
block	o	o	o	o	o
inline	x	x	좌/우	o(설명)	o(설명)
inline-block	o	o	o	o	o

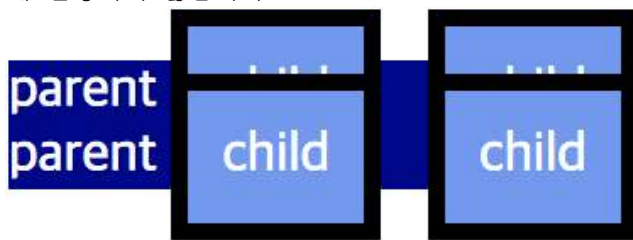
- inline 요소의 padding/border 속성이 좌/우 만 적용 된다고 표시한 이유 추가 설명

실제로 inline 요소의 padding/border는 좌/우뿐만 아니라 상/하에도 적용이 됩니다.



parent는 <div>, child는

하지만, 상/하 padding/border는 line-box에는 영향을 주지 못하기 때문에 위와 같이 부모 요소의 박스에 반영되지 않습니다.



parent는 <div>, child는

또한 인접한 다른 line-box 에도 반영되지않습니다. 즉 콘텐츠가 겹칠 수 있기 때문에 실무에서는 잘 사용하지 않습니다.

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>display</title>
```

```
<style>
```

```

body>div {
  padding: 5px;
  border: 1px dashed #aaa;
}

.box {
  padding: 15px;
  background-color: #eee;
  border: 1px solid #aaa;
}

.none .box {
  display: none;
}

.inline .box {
  display: inline;
}

.block .box {
  display: block;
}

.inline-block .box {
  display: inline-block;
}

.list-item .box {
  display: list-item;
}
</style>
</head>

<body>
  <h1>display</h1>
  <h2>none</h2>
  <div class="none">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
  </div>
  <h2>inline</h2>
  <div class="inline">
    <div class="box">box1</div>

```

```

    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box4</div>
    <div class="box">box5</div>
    <div class="box">box6</div>
    <div class="box">box7</div>
    <div class="box">box8</div>
    <div class="box">box9</div>
    <div class="box">box10</div>
</div>
<h2>block</h2>
<div class="block">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
</div>
<h2>inline-block</h2>
<div class="inline-block">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
</div>
<h2>list-item</h2>
<div class="list-item">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
</div>
</body>
</html>

```

https://www.w3schools.com/cssref/pr_class_display.asp

<https://developer.mozilla.org/en-US/docs/Web/CSS/display>

2) 속성 - visibility

- visibility 속성 : 요소의 화면 표시 여부를 지정하는 속성입니다.

기본값 : visible

visibility: visible | hidden | collapse | initial | inherit;

<속성값>

visible : 화면에 표시

hidden : 화면에 표시되지 않음(공간은 차지함)

collapse : 셀 간의 경계를 무시하고 숨김(테이블 관련 요소에만 적용 가능)

visibility: visible; /* 보임 기본값 */

visibility: hidden; /* 숨김, 자신의 박스 영역은 유지(margin까지 모두 포함) */

visibility: collapse; /* 셀간의 경계를 무시하고 숨김(박스영역 없음, 테이블의 행과 열 요소에만 지정 가

능, 그 외 요소 지정은 hidden과 같음) */

- display: none과 차이점

display: none: 요소가 렌더링 되지 않음(DOM에 존재하지 않음)

visibility: hidden: 요소가 보이지는 않지만 렌더링 되며 화면에 공간을 가지고 있음(DOM에 존재함)

<코드실습> https://www.w3schools.com/cssref/playit.asp?filename=playcss_visibility_collapse

https://www.w3schools.com/cssref/pr_class_visibility.asp

<https://developer.mozilla.org/en-US/docs/Web/CSS/visibility>

3) 속성 - float

- float 속성 : 요소를 float(요소를 보통의 흐름에서 벗어나게 함) 시킬지 지정하는 속성입니다.

기본값 : none

float: none | left | right | initial | inherit;

<속성값>

none : float 시키지 않음(기본값)

left : 좌측으로 float 시킴

right : 우측으로 float 시킴

요소를 보통의 흐름에서 벗어나 띄어지게 함

주변 텍스트나 인라인 요소가 주위를 감싸는 특징이 있음

Lorem ipsum dolor sit amet consectetur
adipiscing elit. Eligendi inventore nisi
non dolorem optio, laudantium nesciunt perspiciatis
consequatur minus quis necessitatibus doloremque
deserunt ullam beatae corrupti provident dolor sed
cumque?

float: right

대부분 요소의 display 값을 block으로 변경함 (display 값 변경 예외: inline-table, flex 등)

▼ display

inline

span

block

<style>...</style>

► float

right

<코드실습>

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8">

<title>float</title>

<style>

.container {

border: 1px dashed #aaa;

padding: 15px;

clear: both;

}

.container div, .container span {

width: 100px;

```

height: 100px;
border: 1px solid #aaa;
color: #fff;
}
.container :nth-child(1) {
background-color: green;
}
.container :nth-child(2) {
background-color: skyblue;
}

```

</style>

</head>

<body>

<h2>요소를 보통의 흐름에서 벗어나 띄워지게 함</h2>

```

<div class="container" style="width:400px;">
  <div style="float:left;">Box1</div>
  <div style="float:left;">Box2</div>
</div>

```

<h2 style="margin-top:100px;">주변 텍스트나 인라인요소가 주위를 감싸는 특징이 있음.</h2>

```

<div class="container" style="width:400px;">
  <div style="float:left;">Box1</div>
  <div style="float:right;">Box2</div>

```

<p>CSS 속성(property) float 은 한 요소(element)가 보통 흐름(normal flow)으로부터 빠져 텍스트 및 인라인(inline) 요소가 그 주위를 감싸는 자기 컨테이너의 좌우측을 따라 배치되어야 함을 지정합니다. 부동(floating) 요소 는 float 의 계산값(computed value)이 none이 아닌 요소입니다.</p>

</div>

<h2>대부분의 요소에 display 값을 block으로 변경함.</h2>

```

<div class="container" style="width:400px;">
  <span style="float:left;">inline1</span>
  <span>inline2</span>

```

</div>

</body>

</html>

https://www.w3schools.com/cssref/pr_class_float.asp

https://www.w3schools.com/cssref/pr_class_float.asp

4) 속성 - clear

- clear 속성 : 요소를 floating 된 요소의 영향에서 벗어나게 하는 속성입니다.

기본값 : none

clear: none | left | right | both | initial | inherit;

<속성값>

none : 양쪽으로 floating 요소를 허용(기본값)

left : 왼쪽으로 floating 요소를 허용하지 않음

right : 오른쪽으로 floating 요소를 허용하지 않음

both : 양쪽으로 floating 요소를 허용하지 않음

block-level 요소만 적용 가능

<코드실습>

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8">

<title>clear</title>

<style>

.container {

border: 1px dashed #aaa;

}

div,

span {

padding: 20px;

border: 1px solid red;

}

</style>

</head>

<body>

<h2>clear: none</h2>

<div class="container">

<div style="float:left">float:left;</div>

내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘
대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로
내용 맘대로 내용 맘대로 내용 맘대로

</div>

<h2>clear: left</h2>

<div class="container">

<div style="float:left">float:left;</div>

내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대
로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로
내용 맘대로 내용 맘대로 내용 맘대로

</div>


```

<h2>clear: right</h2>
<div class="container">
  <div style="float:right">float:right;</div>
  <span style="display:inline-block;clear:right;">내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로
내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용
맘대로 내용 맘대로 내용 맘대로 내용 맘대로</span>
</div>

<h2>clear: both</h2>
<div class="container">
  <div style="float:left">float:left;</div>
  <div style="float:right">float:right;</div>
  <span style="display:block;clear:both;">내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘
대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로 내용 맘대로 내용 맘대로 내용 맘대로내용 맘대로
내용 맘대로 내용 맘대로 내용 맘대로</span>
</div>
</body>
</html>

```

https://www.w3schools.com/cssref/pr_class_clear.asp
<https://developer.mozilla.org/en-US/docs/Web/CSS/clear>

5) 속성 - position

- position 속성 : 요소의 위치를 정하는 방법을 지정하는 속성입니다.

기본값 : static

position: static | absolute | fixed | relative | sticky | initial | inherit;

<속성값>

static : Normal-flow 에 따라 배치되며 offset 값이 적용되지 않는다. (기본값)

absolute : 부모 요소의 위치를 기준으로 offset 에 따라 배치된다. 부모가 position 값(static 제외)을 가지면 offset 값의 시작점이 된다. * 부모의 position 값이 static 인 경우 조상의 position 값이 static이 아닐 때까지 거슬러 올라가 기준으로 삼습니다. Normal-flow의 흐름에서 벗어난다.

fixed : 뷰포트(브라우저의 창)를 기준으로 offset 에 따라 배치된다. 즉, 화면 스크롤에 관계없이 항상 화면의 정해진 위치에 정보가 나타난다. 부모의 위치에 영향을 받지 않는다.

relative : 자신이 원래 있어야 할 위치를 기준으로 offset 에 따라 배치된다. 부모의 position 속성에 영향을 받지 않는다. Normal -flow의 흐름에 따른다. 주변 요소에 영향을 주지 않으면서 offset 값으로 이동한다.

Normal-flow란? 일반적인 상황에서 각의 요소들의 성질에 따라 배치 되는 순서(흐름)를 뜻합니다. 예를 들면, block 레벨 요소들은 상하로 배치되고, inline 레벨 요소들은 좌우로 배치되는 것을 말합니다.

- offset(top/left/bottom/right)

top|bottom|left|right: auto|length|initial|inherit;

top: 50%;

left: 10px;

bottom: -10px;

right: auto;

offset의 % 단위 사용 : 이전에 padding과 margin에서 % 값을 적용할 때, 상하좌우 방향에 관계없이

가로 사이즈를 기준으로 % 값을 계산된다고 배웠습니다. 그러나 offset은 top, bottom (상하)는 기준이 되는 요소의 height 값 left, right (좌우)는 width값에 대하여 계산 되어집니다.

<코드실습>

```
<!DOCTYPE html>
```

```
<html lang="ko">
```

<head>

```
<meta charset="UTF-8">
```

<title>position</title>

<style>

```
.sibling {
```

padding: 5px;

```
background-color: #eee;
```

```
text-align: center;
```

}

```
.parent {
```

border: 1px dashed #aaa;

padding: 10px;

$$\}$$

.child,

```
.child_offset {
```

```
width: 60px;
```

```
height: 60px;
```

padding: 20px;

```
background-color: #dc3636;
```

```
text-align: center;
```

```
color: #fff;
```

```
font-weight: bold;
```

border: 1px solid #333;

}

```
.child_offset {
```

```
background-color: #3677dc;
```

}

```
.static {
```

```
position: static;
```

}

```
.relative {
```

```
position: relative;
```

}

```

.static {
    position: static;
}

.absolute {
    position: absolute;
}

.fixed {
    position: fixed;
}
</style>
</head>
<body>
<h1>position (Offset 미지정)</h1>
<h2>static (기본)</h2>
<div class="parent">
    <div class="sibling">Sibling 1</div>
    <div class="child static">static</div>
    <div class="sibling">Sibling 2</div>
</div>

<h2>relative</h2>
<div class="parent">
    <div class="sibling">Sibling 1</div>
    <div class="child relative">relative</div>
    <div class="sibling">Sibling 2</div>
</div>

<h2>absolute</h2>
<div class="parent">
    <div class="sibling">Sibling 1</div>
    <!-- absolute, fixed는 inline요소일때 display:block으로 변경시킴. inline-block 값을 임의 지정했
을 때는 그 값을 유지함. -->
    <span class="child absolute">absolute</span>
    <div class="sibling">Sibling 2</div>
</div>

<h2>fixed</h2>
<div class="parent">
    <div class="sibling">Sibling 1</div>
    <!-- 현재 fixed는 offset값이 없어 뷰포트 밖으로 밀려나오면서 화면 아래로 사라짐 -->
    <div class="child fixed">fixed</div>
    <div class="sibling">Sibling 2</div>

```

</div>

<h1>position (Offset 지정)</h1>

<h2>static - </h2>

<div class="parent">

<div class="sibling">Sibling 1</div>

<div class="child_offset static" style="top: 40px; right: 40px;">static
top: 40, left: 40</div>

<div class="sibling">Sibling 2</div>

</div>

<h2>relative</h2>

<div class="parent">

<div class="sibling">Sibling 1</div>

<div class="child_offset relative" style="top: 40px; left: 40px;">relative
top: 40, left: 40</div>

<div class="sibling">Sibling 2</div>

</div>

<h2>absolute - parent relative</h2>

<div class="parent relative">

<div class="sibling">Sibling 1</div>

<div class="child_offset absolute" style="top: 40px; left: 80px;">absolute
top: 40, left: 80</div>

<div class="sibling">Sibling 2</div>

</div>

<h2>fixed</h2>

<!-- 부모가 fixed여도 별개로 위치함 -->

<div class="parent">

<div class="sibling">Sibling 1</div>

<div class="child_offset fixed" style="top:100px;left:80%;">fixed
top:100, left:80%</div>

<div class="sibling">Sibling 2</div>

</div>

</body>

</html>

https://www.w3schools.com/cssref/pr_class_position.asp

<https://developer.mozilla.org/en-US/docs/Web/CSS/position>

6) 속성- z-index

- z-index 속성 : 요소가 겹치는 순서(쌓임 순서 또는 stack order)를 지정하는 속성입니다.

기본값 : auto

z-index: auto | number | initial | inherit;

<속성값>

auto : 쌓임 순서를 부모와 동일하게 설정(기본값)

number : 해당 수치로 쌓임 순서를 설정(음수 허용)

z-index: 1;

position 값이 static이 아닌 경우 지정가능

순서 값이 없을 경우 생성순서(코드상 순서)에 따라 쌓임

부모가 z-index 값이 있을 경우 부모 안에서만 의미있음

큰 값이 가장 위쪽(음수 사용 가능)

<코드실습>

```

<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>z-index</title>
  <style>
    .parent {
      z-index: 10;
      position: relative;
      width: 300px;
      height: 50px;
      border: 2px solid #000;
      background-color: #ccc;
    }

    .child {
      z-index: 10;
      position: absolute;
      top: 10px;
      right: 10px;
      width: 100px;
      height: 100px;
      background-color: pink;
      border: 2px solid red;
    }
  </style>
</head>
<body>
  <div class="wrap">

    <h1 class="practive_title">z-index 속성</h1>
    <div class="z_area">
      <div class="parent" style="z-index:11">
        position: relative;
        <div class="child" style="right:20px;z-index:1000;">

```

```
        position: absolute;
    </div>
</div>
<div class="parent" style="">
    position: relative;
    <div class="child">
        position: absolute;
    </div>
</div>
</div>
</body>
</html>
```

<https://codepen.io/yongwon/pen/dXwyQq>

https://www.w3schools.com/cssref/playit.asp?filename=playcss_z-index&preval=2

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index

7) HTML/CSS 유효성 검사

<https://validator.w3.org/>

8. 미디어쿼리

1) 미디어쿼리 소개 : 미디어쿼리(Media Queries)는 각 미디어 매체에 따라 다른 스타일(css style)을 적용할 수 있게 만드는 것입니다. 미디어 매체는 모니터와 같은 스크린 매체, 프린트, 스크린 리더기와 같은 것들을 이야기 합니다. 미디어쿼리는 동일한 웹 페이지를 다양한 환경의 사용자들에게 최적화된 경험을 제공할 수 있게 해줍니다. CSS2의 미디어 타입(Media Types)을 확장해서 만들어졌습니다. 미디어타입은 이론적으로는 훌륭했지만, 결과적으로 제대로 활용되지 못 했습니다. 당시에는 미디어 타입을 제대로 지원하는 기기가 없었기 때문입니다. 미디어쿼리가 등장하기 이전에는 제대로 된 반응형 웹 사이트를 제작할 수는 없었습니다. 하지만 당시에는 사용자들의 환경이 아주 제한적이었기 때문에 제작자 입장에서는 대중적인 미디어 범위에서만 잘 보이도록 사이트를 제작하면 반응형이 아니더라도 충분했습니다. 하지만 웹이 급격히 발전하면서 대응해야 하는 미디어의 폭이 상당히 늘어났습니다. 이런 필요성에 따라 W3C는 CSS2의 미디어 타입을 확장하여, CSS3 미디어쿼리를 발표합니다. 이 미디어쿼리로 인해 웹 사이트를 제작함에 있어 이전의 정적인 고정 레이아웃 웹 사이트에서 동적으로 반응하는 반응형 웹 사이트로 패러다임이 새롭게 변화하였습니다.

<https://www.w3.org/TR/1998/WD-css2-19980128/media.html>

<https://www.w3.org/TR/css3-mediaqueries/>

2) 미디어 타입 & 미디어 특성

- @media(at media) : 미디어쿼리는 CSS2 Media Types을 확장했기 때문에 선언 방법은 동일합니다.

@media mediaqueries { /* style rules */ }

@media 로 시작하며, 이 키워드는 이제부터 미디어쿼리를 시작한다 라는 뜻입니다. 그 뒤에 미디어쿼리 구문(위 코드의 mediaqueries) 이 나오고 이어서 중괄호({ })를 이용해서 스타일 규칙이 들어갑니다. 미디어쿼리 구문은 논리적으로 평가되며 참이면 뒤에 나오는 스타일 규칙이 적용되고, 거짓이면 무시됩니다. 미디어쿼리 구문은 미디어 타입(Media Types)과 미디어 특성(Media Features)으로 이루어져 있습니다.

- 미디어 타입 : all, braille, embossed, handheld, print, projection, screen, speech, tty, tv

우리가 알아야 할 타입은 all, print, screen 정도입니다. 그 중에서도 screen이 거의 대부분입니다. 화면을 출력하는 디스플레이가 있는 미디어들은 전부 screen에 속하기 때문에 현실적으로 고려해야 하는 미디어들은 전부 여기에 해당이 됩니다. print 타입도 간혹 사용이 됩니다. 실습할 때 다룹니다. all 타입은 모든 미디어에 적용되는 타입입니다. 미디어를 구분하는 용도가 아니기 때문에 유용하게 사용되지는 않습니다.

- 미디어 특성 : width, height, device-width, device-height, orientation, aspect-ratio, device-aspect-ratio, color, color-index, monochrome, resolution, scan, grid

미디어 특성 역시 우리가 알아야 할 특성은 width와 orientation 정도입니다. width는 뷰포트의 너비, 즉, 브라우저 창의 너비를 말합니다.(스크린의 크기 x) orientation은 미디어가 세로모드인지 가로모드인지를 구분합니다. 미디어쿼리에서는 이 구분을 width와 height 특성의 값을 비교해서 height가 width보다 같거나 크면 세로모드 반대의 경우에는 가로모드라고 해석합니다. 세로모드에서는 portrait, 가로모드에서는 landscape 키워드와 매칭이 됩니다.

- 미디어쿼리 level 4 : 우리 수업의 미디어 타입과 미디어 특성은 CSS3 미디어쿼리 표준 명세를 기준으로 작성되었습니다. 현재, 미디어쿼리 level 4가 CR(유력 후보안) 단계입니다. 해당 문서에서 미디어 타입 대부분과 미디어 특성 중 일부 속성이 폐기 예정입니다.

<https://www.w3.org/TR/css3-mediaqueries/#media1>

<https://www.w3.org/TR/mediaqueries-4/#media-types>

<https://www.w3.org/TR/mediaqueries-4/#mf-deprecated>

3) Syntax

- 미디어쿼리 Syntax

media_query_list

: S* [media_query [',' S* media_query]*]?

;

media_query

: [ONLY | NOT]? S* media_type S* [AND S* expression]*

| expression [AND S* expression]*

;

expression

: '(' S* media_feature S* [':' S* expr]? ')' S*

;

위 코드는 CSS3 미디어쿼리 표준 명세에 나와 있는 Syntax 부분입니다. 우리가 알아야 할 부분만 적어 놓았습니다. 참고로 Syntax는 전부 이해할 필요는 없지만, 일부 기호는 알아두면 좋습니다.

[a] : a가 나올 수도 있고 나오지 않을 수도 있습니다.

a | b : a 또는 b 둘 중에 하나를 선택합니다.

"\" : 파이프 라인 기호로 키보드의 역슬래시(\) 키를 Shift 키를 누른 채로 누르면 나옵니다.

a? : a가 0번 나오거나 1번만 나올 수 있음

a* : a가 0번 나오거나 그 이상 계속 나올 수 있음

media_type : all, screen, print 등 명세에 정의된 미디어 타입

media_feature : width, orientation 등 명세에 정의된 미디어 특성

media_query_list : 여러 개의 미디어쿼리로 이루어진 리스트로 작성 가능하며, 쉼표를 이용해서 구분합니다.

media_query : A 형태 - 미디어 타입에 and 키워드를 이용해서 미디어 표현식을 붙일 수 있습니다.

미디어 타입 앞에는 only 또는 not 키워드가 올 수 있습니다.

미디어 표현식은 생략 가능하기 때문에 미디어 타입 단독으로 사용될 수 있습니다.

: B 형태 - 미디어 타입 없이 미디어 표현식이 바로 나올 수 있습니다.(미디어 타입이 명시되지 않으면 all로 간주합니다.) 미디어 표현식은 and 키워드를 이용해서 계속해서 나올 수 있습니다.

expression : 미디어 표현식은 괄호로 감싸야 하며, 특성 이름과 해당하는 값으로 이루어집니다. 이름과 값은 : 기호로 연결합니다. 또, 값이 없이 특성 이름만으로도 작성할 수 있다.

- min-/max- 접두사 : 미디어 특성은 이름 앞에 min- 또는 max- 접두사를 붙일 수 있습니다. 실제로 반응형 사이트를 제작할 때는 보통 접두사를 붙여서 사용합니다. 접두사를 붙이지 않고 사용하는 경우 대부분 효율적이지 못하기 때문입니다. 예를 들어 대부분의 반응형 사이트는 width 특성을 이용하는데, 접두사 없이 width: 00px 이라고 하게 선언하면 정확히 뷰포트의 크기가 00px 에서만 적용되기 때문에, 다양한 기기들을 대응하기 힘듭니다. 그래서 접두사를 사용하여 범위를 지정하게 되면 훨씬 간결하게 반응형 사이트를 제작할 수 있습니다.

- 예제 코드 : 위에서 정의한 Syntax 따라 유효한 미디어쿼리 예제 코드를 살펴보고 어떻게 해석이 되는지

@media (min-width: 768px) and (max-width: 1024px) { ... } : and는 연결된 모든 표현식이 참이면 적용됩니다.(and 키워드는 연결된 부분이 모두 참이어야 적용이 됩니다.)

@media (color-index) : 미디어 장치가 color-index를 지원하면 적용됩니다.

@media screen and (min-width: 768px), screen and (orientation: portrait), ... : 쉼표로 연결된 미디어쿼리 중 하나라도 참이면 적용됩니다.(and 키워드와 반대라고 생각하면 됩니다.)

@media not screen and (min-width: 768px)

: not 키워드는 하나의 media_query 전체를 부정합니다.

: (not screen) and (min-width: 768px) 잘못된 해석!

: not (screen and (min-width: 768px)) 올바른 해석!

: @media not screen and (min-width: 768px), print

첫 번째 미디어쿼리에만 not 키워드가 적용되며, 두 번째 미디어쿼리(print)에는 영향이 없습니다.

- 미디어쿼리 선언 방법

@media screen and (color) : CSS 파일 내부에 또는 <style> 태그 내부에 사용가능 합니다. 대부분의 경우 이렇게 사용합니다.

<link rel="stylesheet" media="screen and (color)" href="example.css"> : <link> 태그의 media 속성에 미디어쿼리를 선언합니다. 미디어쿼리가 참이면 뒤에 css 파일 규칙이 적용됩니다.

@import url(example.css) screen and (color); : CSS 파일 내부에 또는 <style> 태그 내부에 사용가능 합니다. @import문 뒤에 미디어쿼리를 선언하면 됩니다.

4) 실습

- 실습 1. 디스플레이 크기에 따른 body요소의 background-color 변경 : 0~767px 이면 : gold, 768px~1024px 이면 : lightblue, 1025px~ 이면 : lightpink

```
@media (max-width: 767px) {  
    body { background-color: gold; }  
}  
@media (min-width: 768px) and (max-width: 1024px) {  
    body { background-color: lightblue; }  
}  
@media (min-width: 1025px) {  
    body { background-color: lightpink; }  
}
```

위 코드의 경우, 3개의 조건을 만족하기 위해 3개의 미디어쿼리를 작성했습니다. 위 코드도 우리가 의도한 대로 정상적으로 잘 동작하지만, 1개의 조건을 기본 body 요소의 배경 색상으로 지정해놓게되면 다른 2개의 미디어쿼리만 이용해도 가능합니다. 어떤 조건을 기본으로 정할지는 작성하는 사이트가 모바일 사이트인지, 데스크탑 사이트인지를 먼저 구분해야 합니다. 모바일 중심의 사이트라면(모바일 first) 모바일에 해당하는 조건의 배경 색상을 기본으로 선언하면 되고, 데스크탑 중심의 사이트라면(데스크탑 first) 데스크탑에 해당하는 조건의 배경 색상을 기본으로 선언해놓으면 됩니다. 만들어진 미디어쿼리를 수정해서, 모바일 first인 경우 또 데스크탑 first인 경우대로 한 번씩 작성을 해보겠습니다.

```
mobile first  
/* mobile first*/  
body { background-color: gold; }  
@media (min-width: 768px) and (max-width: 1024px) {  
    body { background-color: lightblue; }  
}  
@media (min-width: 1025px) {  
    body { background-color: lightpink; }  
}  
  
desktop first  
/* desktop first*/  
body { background-color: lightpink; }  
@media (min-width: 768px) and (max-width: 1024px) {
```

```
body { background-color: lightblue; }
}
```

```
@media (max-width: 767px) {
  body { background-color: gold; }
}
```

- 실습2. 웹 페이지를 인쇄하는 경우의 스타일 추가 : 앵커 요소의 url 출력, 앵커 요소의 밑줄 제거
미디어 타입 print를 이용하면, 인쇄될 경우에 적용되는 스타일을 추가할 수 있습니다. 먼저 앵커 요소의 url 을 텍스트 뒤에 붙여서 나타나게 하고, 링크임을 표시해주는 밑줄도 제거를 하겠습니다. 웹 페이지를 인쇄할 경우에는 앵커 요소가 가리키는 url을 문서에 같이 출력해주는게 내용을 이해하는데 훨씬 좋습니다. 실제 인쇄를 하지 않더라도 브라우저에서 제공하는 인쇄 미리보기 기능을 이용하면 화면으로 확인이 가능합니다.

```
@media print {
  a { text-decoration: none; }
  a:after { display: inline; content: '(' attr(href) ')'; }
}
```

<코드실습>

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8">

<title>Media Queries</title>

<style>

/* 1 */

```
@media (max-width: 767px) {
  body { background-color: gold; }
}
```

```
@media (min-width: 768px) and (max-width: 1024px) {
  body { background-color: lightblue; }
}
```

```
@media (min-width: 1025px) {
  body { background-color: lightpink; }
}
```

/* 2 mobile first*/

```
body { background-color: gold; }
```

```
@media (min-width: 768px) {
  body { background-color: lightblue; }
}
```

```
@media (min-width: 1025px) {
  body { background-color: lightpink; }
}
```

/* 3 desktop first*/

```
body { background-color: lightpink; }
```

```

    @media (max-width: 1024px) {
        body { background-color: lightblue; }
    }
    @media (max-width: 767px) {
        body { background-color: gold; }
    }

    /* 4 print */
    @media print {
        a { text-decoration: none; }
        a:after { display: inline; content: '(' attr(href) ' '; }
    }
</style>
</head>
<body>
    <p>W3C는 <a href="https://www.w3.org/TR/css3-mediaqueries/">CSS3 미디어쿼리 문서</a>를 2012년 6월에 표준 권고안으로 제정하였습니다.</p>
    <p>또한, 기존의 미디어쿼리 개선 작업을 진행 중이며, 2017년 9월에 <a href="https://www.w3.org/TR/mediaqueries-4/">미디어쿼리 레벨4</a>를 발표했습니다. 이 문서는 현재 유력 표준 권고안입니다.</p>
</body>
</html>

```

<https://www.w3.org/TR/mediaqueries-4/#mq-min-max>

+ 추가. Viewport 이해

- 뷰포트 설정 : 뷰포트를 설정하는 태그는 <meta> 태그로 <head> 태그에 위치해야 합니다. name 속성에 "viewport"라고 선언하며 content 속성에 뷰포트를 설정하는 내용이 들어갑니다.

```
<meta name="viewport" content="뷰포트의 설정 값" >
```

content 설정

width(height) : 뷰포트의 가로(세로) 크기를 지정합니다. px단위의 수치가 들어갈 수 있지만, 대부분 특수 키워드인 "device-width(height)"를 사용합니다.(뷰포트의 크기를 기기의 스크린 width(height) 크기로 설정한다는 의미입니다.)

initial-scale : 페이지가 처음 나타날 때 초기 줌 레벨 값을 설정합니다.(소수값)

user-scalable : 사용자의 확대/축소 기능을 설정할 수 있습니다.

모바일 웹 사이트의 뷰포트 설정

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

https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag

https://www.w3schools.com/css/css_rwd_viewport.asp