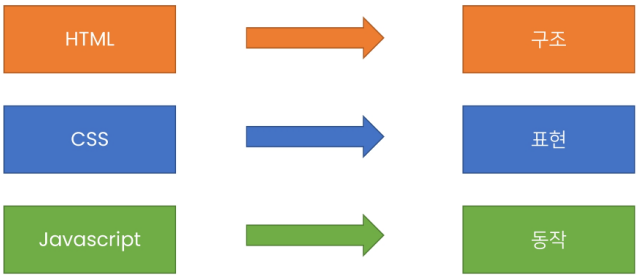
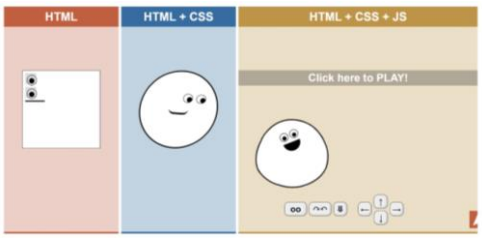
<목차>

<Web이란?>

- **웹 사이트의 구성 요소**

- 웹 사이트란 브라우저를 통해서 접속하는 웹 페이지(문서)들의 모음
- 웹 페이지는 글, 그림, 동영상 등 여러가지 정보를 담고 있으며, 마우스로 클릭하면 다른 웹 페이지로 이동하는 '링크'들이 있음. '링크'를 통해 여러 웹 페이지를 연결한 것을 웹 사이트라고 함.





- **웹 사이트와 브라우저**

- 웹 사이트는 브라우저를 통해 동작함
- 브라우저마다 동작이 약간씩 달라서 문제가 생기는 경우가 많음(파편화)
- 해결책으로 웹 표준이 등장

- **웹 표준**

- 웹에서 표준적으로 사용되는 기술이나 규칙
- 어떤 브라우저든 웹 페이지가 동일하게 보이도록 함(크로스 브라우징)

<개발 환경 설정>

- **Visual Studio Cod**

: HTML / CSS 코드 작성을 위한 Visual Studio Code 추천 확장 프로그램

- Open in browser
- Auto rename tag
- Highlight Matching Tag

- **크롬 개발자 도구**

: 웹 브라우저 크롬에서 제공하는 개발과 관련된 다양한 기능을 제공

: 주요 기능

- Elements – DOM 탐색 및 CSS 확인 및 변경
  - Styles – 요소에 적용된 CSS 확인
  - Computed – 스타일이 계산된 최종 결과
  - Event Listeners – 해당 요소에 적용된 이벤트(JS)
- Sources, Network, Performance, Application, Security, Audits 등

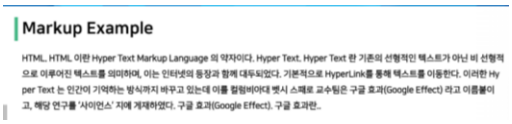
<HTML>

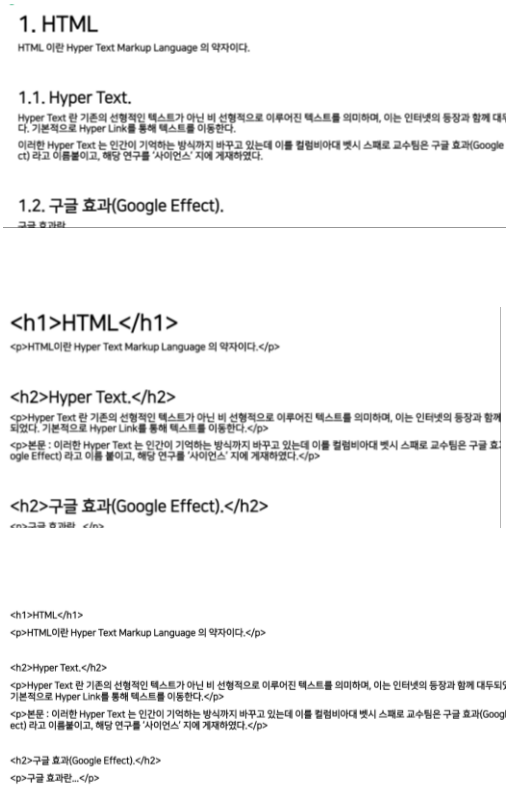
: Hyper Text Markup Language

: 참조(하이퍼링크)를 통해 사용자가 한 문서에서 다른 문서로 즉시 접근할 수 있는 텍스트

: 태그 등을 이용하여 문서나 데이터의 구조를 명시하는 언어

Ex) HTML, Markdown





- **HTML이란?**

: 웹 페이지를 작성(구조화)하기 위한 언어

.html
HTML 파일

- **HTML 스타일 가이드**

```
<body>
  <h1> 웹문서 </h1>
  <ul>
    <li>HTML</li>
    <li>CSS</li>
  </ul>
</body>
```

마크업 스타일 가이드(2 space)

[HTML의 기본 구조]

1. **html:** 문서의 최상위(root) 요소
2. **head**: 문서 메타데이터 요소
   - 문서 제목, 인코딩, 스타일, 외부 파일 로딩 등
   - 일반적으로 브라우저에 나타나지 않는 내용
3. **body:** 문서 본문 요소
   - 실제 화면 구성과 관련된 내용

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>

</body>
</html>
```

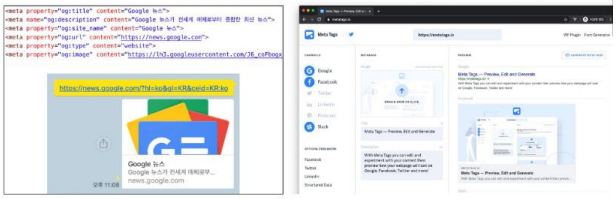
- **head 예시**

- **<title>** : 브라우저 상단 타이틀
- **<meta>** : 문서 레벨 메타데이터 요소
- **<link> :** 외부 리소스 연결 요소 (CSS 파일, favicon 등)
- **<script>** : 스크립트 요소 (JavaScript 파일 / 코드)
- **<style> :** CSS 직접 작성

```
<head>
  <title>HTML 수업</title>
  <meta charset="UTF-8">
  <link href="style.css" rel="stylesheet">
  <script src="javascript.js"></script>
  <style>
    p {
      color: black;
    }
  </style>
</head>
```

head 예시 : Open Graph Protocol

- 메타 데이터를 표현하는 새로운 규약
  - HTML 문서의 메타 데이터를 통해 문서의 정보를 전달
  - 메타정보에 해당하는 제목, 설명 등을 쓸 수 있도록 정의



- **요소(element)**

(여는/시작)태그 (닫는/종료)태그

<h1>contents</h1>

HTML의 요소는 태그와 내용(contents)로 구성되어 있다.

- *HTML 요소는 시작 태그와 종료 태그 그리고 태그 사이에 위치한 내용으로 구성*
  - 요소는 태그로 컨텐츠(내용)을 감싸는 것으로 그 정보의 성격과 의미를 정의

- 내용이 없는 태그들도 존재(닫는 태그가 없음)

  : br, hr, img, input, link, meta

- *요소는 중첩(nested)될 수 없음*
  - 요소의 중첩을 통해 하나의 문서를 구조화
  - 여는 태그와 닫는 태그의 쌍을 잘 확인해야함
    - 오류를 반환하는 것이 아닌 그냥 레이아웃이 깨진 상태로 출력되기 때문에, 디버깅이 힘들어 질 수 있음

- **속성**

```
<a href="https://google.com"></a>
   속성명          속성값
```

태그별로 사용할 수 있는 속성은 다르다.

```
<a href="https://google.com"></a>
   공백은 No!                 ""(쌍따옴표) 사용!
```

속성 지정 스타일 가이드

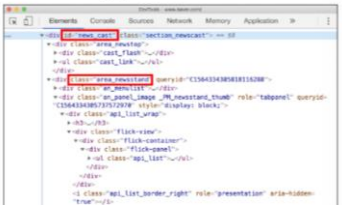
- 속성을 통해 태그의 부가적인 정보를 설정할 수 있음
- 요소는 속성을 가질 수 있으며, 경로나 크기와 같은 추가적인 정보를 제공
- 요소의 시작 태그에 작성하며 보통 이름과 값이 하나의 쌍으로 존재
- 태그와 상관없이 사용 가능한 속성(HTML Global Attribute)들도 있음

- **HTML Global Attribute**

: 모든 HTML 요소가 공통으로 사용할 수 있는 대표적인 속성 (몇몇 요소에는 아무 효과가 없을 수 있음)

- **id :** 문서 전체에서 유일한 고유 식별자 지정
- **class :** 공백으로 구분된 해당 요소의 클래스의 목록 (CSS, JS에서 요소를 선택하거나 접근)
- **data-* :** 페이지에 개인 사용자 정의 데이터를 저장하기 위해 사용
- **style :** 요소에 대한 추가 정보 지정
- **tabindex :** 요소의 탭 순서

HTML Global Attribute 예시



HTML 코드 예시

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <!-- 이것은 주석입니다. -->
  <h1>나의 첫번째 HTML</h1>
  <p>이것은 본문입니다.</p>
  <span>이것은 인라인요소</span>
  <a href="https://www.naver.com">네이버로 이동!!</a>
</body>
</html>
```

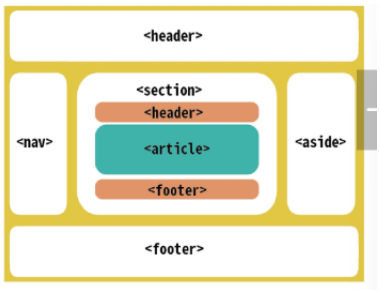
- **시맨틱 태그**

: HTML5에서 의미론적 요소를 담은 태그의 등장

➔ 기존 영역을 의미하는 div태그를 대체하여 사용

: 대표적인 태그 목록

- **header :** 문서 전체나 섹션의 헤더(머리말 부분)
- **nav :** 내비게이션
- **aside :** 사이드에 위치한 공간, 메인 콘텐츠와 관련성이 적은 콘텐츠
- **section :** 문서의 일반적인 구분, 컨텐츠의 그룹을 표현
- **article :** 문서, 페이지, 사이트 안에서 독립적으로 구분되는 영역
- **footer :** 문서 전체나 섹션의 푸터(마지막 부분)




```
<div>
  <div></div>
</div>
<div>
  <div></div>
  <div></div>
</div>
<div></div>
```

```
<header>
  <nav></nav>
</header>
<section>
  <article></article>
  <article></article>
</section>
<footer></footer>
```

- Non semantic 요소는 div, span 등이 있으며

  h1, table 태그들도 시맨틱 태그로 볼 수 있음

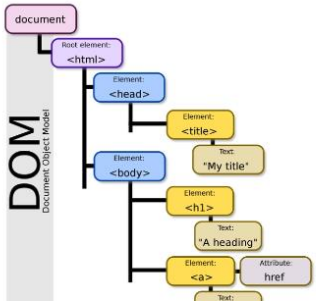
- 개발자 및 사용자 뿐만 아니라 검색엔진 등에 의미 있는 정보의 그룹을 태그로 표현
- 단순히 구역을 나누는 것 뿐만 아니라 '의미'를 가지는 태그들을 활용하기 위한 노력
- 요소의 의미가 명확해지기 때문에 코드의 가독성을 높이고 유지보수를 쉽게 함
- 검색엔진최적화(SEO)를 위해서 메타태그, 시맨틱 태그 등을 통한 마크업을 효과적으로 활용 해야함

- **렌더링**

: 웹사이트 코드를 사용자가 보게 되는 웹 사이트로 바꾸는 과정

- **DOM(Document Object Model) 트리**

- 텍스트 파일인 HTML 문서를 브라우저에서 렌더링 하기 위한 구조
  - HTML 문서에 대한 모델을 구성함
  - HTML 문서 내의 각 요소에 접근 / 수정에 필요한 프로퍼티와 메서드를 제공함




```
<body>
  <h1> 웹문서 </h1>
  <ul>
    <li>HTML</li>
    <li>CSS</li>
  </ul>
</body>
```

```
<body>
  <h1> 웹문서 </h1>
  <ul>
    <li>HTML</li>
    <li>CSS</li>
  </ul>
</body>
```

[HTML 문서 구조화]

- **인라인 / 블록 요소**

- HTML 요소는 크게 인라인 / 블록 요소로 나눔
- 인라인 요소는 글자처럼 취급
- 블록 요소는 한 줄 모두 사용

- **텍스트 요소**

태그	설명
<a>	href 속성을 활용하여 다른 URL로 연결하는 하이퍼링크 생성
 	굵은 글씨 요소 중요한 강조하고자 하는 요소 (보통 굵은 글씨로 표현)
 	기울임 글씨 요소 중요한 강조하고자 하는 요소 (보통 기울임 글씨로 표현)
 	텍스트 내에 줄 바꿈 생성
<img>	src 속성을 활용하여 이미지 표현
	의미 없는 인라인 컨테이너

- **그룹 컨텐츠**

태그	설명
	하나의 문단 (paragraph)
<hr>	문단 레벨 요소에서의 주제의 분리를 의미하며 수평선으로 표현됨 (A Horizontal Rule)
 	순서가 있는 리스트 (ordered) 순서가 없는 리스트 (unordered)
<pre></pre>	HTML에 작성한 내용을 그대로 표현. 보통 고정폭 글꼴이 사용되고 공백문자를 유지
<blockquote> </blockuote>	텍스트가 긴 인용문 주로 들여쓰기를 한 것으로 표현됨
	의미 없는 블록 레벨 컨테이너

- **Form**

: <form>은 정보(데이터)를 서버에 제출하기 위해 사용하는 태그

:<form> 기본 속성

- **action** : form을 처리할 서버의 URL(데이터를 보낼 곳)
- **method** : form을 제출할 때 사용할 HTTP 메서드 (GET 혹은 POST)
- **enctype** : method가 post인 경우 데이터의 유형
  - application/x-www-form-urlencoded : 기본값
  - multipart/form-data : 파일 전송시 (input type이 file인 경우)
  - text/plain: HTML5 디버깅 용 (잘 사용되지 않음)

```
<form action="/search" method="GET">

</form>
```

● input

: 다양한 타입을 가지는 입력 데이터 유형과 위젯이 제공됨

:<input> 대표적인 속성

- **name** : form control에 적용되는 이름(이름/값 페어로 전송됨)
- **value** : form control에 적용되는 값(이름/값 페어로 전송됨)
- **required, readonly, autofocus, autocomplete, disabled**

```
<form action="/search" method="GET">
  <input type="text" name="q">
</form>
```

● input label

- label을 클릭하여 input 자체의 초점을 맞추거나 활성화 시킬 수 있음

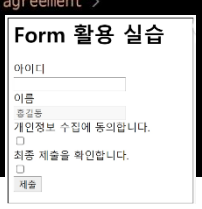
  : 사용자는 선택할 수 있는 영역이 늘어나 웹 / 모바일(터치) 환경에서 편하게 사용할 수 있음

  : label과 input 입력의 관계가 시각적 뿐만 아니라 화면리더기에서도 label을 읽어 쉽게 내용을 확인 할 수 있도록 함

- **<input>에 id 속성을, <label>에는 for 속성을** 활용하여 상호 연관을 시킴

```
<label for="agreement">개인정보 수집에 동의합니다.</label>
<input type="checkbox" name="agreement" id="agreement">
```

```
<body>
  <h1>Form 활용 실습</h1>
    <form action="">
      <!-- autofocus 및 label 확인 -->
      <div class="input-group">
        <label for="username">아이디</label>
      </div>
      <input type="text" name="username" id="username" autofocus>

      <!-- disabled 및 value 확인 -->
      <div class="input-group">
        <label for="name">이름</label>
      </div>
      <input type="text" name="name" value="홍길동" id="name" disabled>

      <!-- label 확인 -->
      <div class="input-group">
        <label for="agreement">개인정보 수집에 동의합니다.</label>
      </div>
      <input type="checkbox" name="agreement" id="agreement">
      <div class="input-group">
        <label>최종 제출을 확인합니다.</label>
      </div>
      <input type="checkbox">
    </form>
    <input type="submit" value="제출">
</body>
```



● Input 유형

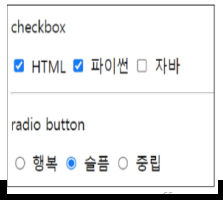
1. 일반

   : 일반적으로 입력을 받기 위하여 제공되는 타입별로 HTML 기본 검증 혹은 추가 속성을 활용할 수 있음

- **text** : 일반 텍스트 입력
- **password** : 입력 시 값이 보이지 않고 문자를 특수기호(*)로 표현
- **email** : 이메일 형식이 아닌 경우 form 제출 불가
- **number** : min, max, step 속성을 활용하여 숫자 범위 설정 가능
- **file** : accept 속성을 활용하여 파일 타입 지정 가능

2. 항목 중 선택

- 일반적으로 label 태그와 함께 사용하여 선택 항목을 작성함
- 동일 항목에 대하여는 name을 지정하고 선택된 항목에 대한 value를 지정해야 함
  - **Checkbox** : 다중 선택
  - **Radio** : 단일 선택



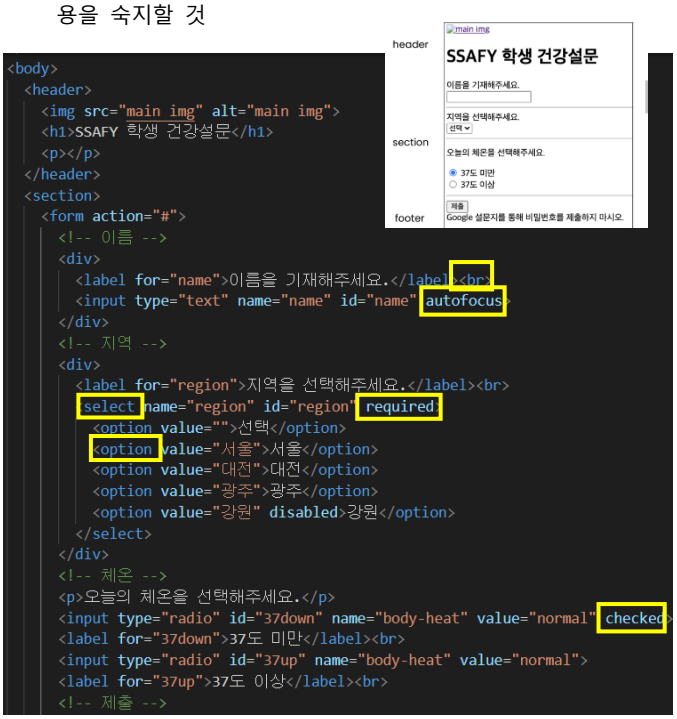

```
<div>
  <p>checkbox</p>
  <input id="html" type="checkbox" name="language" value="html">
  <label for="html">HTML</label>
  <input id="python" type="checkbox" name="language" value="python">
  <label for="python">파이썬</label>
  <input id="python" type="checkbox" name="language" value="java">
  <label for="java">자바</label>
```

3. 기타

- 다양한 종류의 input을 위한 picker를 제공
  - **color** : color picker
  - **date** : date picker
- hidden input을 활용하여 사용자 입력을 받지 않고 서버에 전송되어야 하는 값을 설정
  - **hidden** : 사용자에게 보이지 않는 input

4. 종합

: <input> 요소의 동작은 type에 따라 달라지므로, 각각의 내용을 숙지할 것




```
<body>
  <header>
    <img src="main_img" alt="main img">
    <h1>SSAFY 학생 건강설문</h1>
    <p></p>
  </header>
  <section>
    <form action="#">
      <!-- 이름 -->
      <div>
        <label for="name">이름을 기재해주세요.</label><br>
        <input type="text" name="name" id="name" autofocus>
      </div>
      <!-- 지역 -->
      <div>
        <label for="region">지역을 선택해주세요.</label><br>
        <select name="region" id="region" required>
          <option value="">선택</option>
          <option value="서울">서울</option>
          <option value="대전">대전</option>
          <option value="광주">광주</option>
          <option value="강원" disabled>강원</option>
        </select>
      </div>
      <!-- 체온 -->
      <p>오늘의 체온을 선택해주세요.</p>
      <input type="radio" id="37down" name="body-heat" value="normal" checked>
      <label for="37down">37도 미만</label><br>
      <input type="radio" id="37up" name="body-heat" value="normal">
      <label for="37up">37도 이상</label><br>
      <!-- 제출 -->
      <input type="submit" value="제출">
    </form>
  </section>
  <footer>
    Google 설문지를 통해 비밀번호를 제출하지 마시오.
  </footer>

</body>
</html>
```

<CSS>

: Cascading Style Sheets

: 스타일을 지정하기 위한 언어

: 선택하고, 스타일을 지정한다.

- CSS구문은 선택자를 통해 스타일을 지정할 HTML 요소를 선택
- 중괄호 안에서는 속성과 값, 하나의 쌍으로 이루어진 선언을 진행
- 각 쌍은 선택한 요소의 속성, 속성에 부여할 값을 의미
  - 속성(Property): 어떤 스타일 기능을 변경할지 결정
  - 값(Value): 어떻게 스타일 기능을 변경할지 결정

- CSS 정의 방법

- **인라인**(inline)

  : 인라인을 쓰게 되면 실수가 잦아짐(중복도 있을 것이고, 찾기가 어려워서)

- **내부 참조**(embedding) - **<style>**

  : 내부 참조를 쓰게 되면 코드가 너무 길어짐

- **외부 참조**(link file) – **분리된 CSS 파일**

  : 가장 많이 쓰는 방식

1. 인라인

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1 style="color: blue; font-size: 100px;">Hello</h1>
</body>
</html>
```

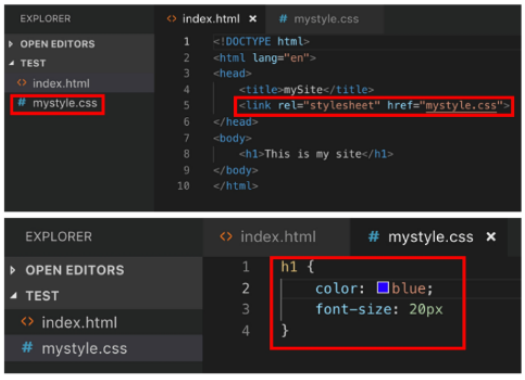
해당 태그에 직접 style 속성을 활용

2. 내부 참조

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    h1 {
      color: blue;
      font-size: 100px;
    }
  </style>
</head>
<body>
</body>
</html>
```

<head> 태그 내에 <style>에 지정

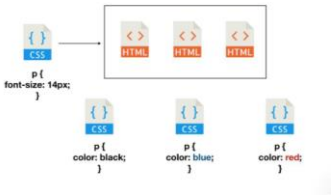
3. 외부 참조



외부 CSS 파일을 <head>내 <link>를 통해 불러오기

CSS 정의 방법 (내부 참조)

CSS 정의 방법 (외부 참조)



[CSS Selectors]

- **선택자(Selector) 유형**

- **기본 선택자**

  : 전체 선택자, 요소 선택자

  : 클래스 선택자, 아이디 선택자, 속성 선택자

- **결합자(Combinators)**

  : 자손 결합자, 자식 결합자

  : 일반 형제 결합자, 인접 형제 결합자

- **의사 클래스/요소(Pseudo Class)**

  : 링크, 동적 의사 클래스

  : 구조적 의사 클래스, 기타 의사 클래스, 의사 엘리먼트, 속성 선택자

- **CSS 선택자 정리**

- **요소 선택자**

  : HTML 태그를 직접 선택

- **클래스(class) 선택자**

  : 마침표(.)문자로 시작하며, 해당 클래스가 적용된 항목을 선택

- **아이디(id) 선택자**

  : # 문자로 시작하며, 해당 아이디가 적용된 항목을 선택

  : 일반적으로 하나의 문서에 1번만 사용

  : 여러 번 사용해도 동작하지만, 단일 id를 사용하는 것을 권장

- **CSS 적용 우선순위 (cascading order)**

1. 중요도 (Importance) – 사용시 주의

   : !important

2. 우선 순위 (Specificity)

   : 인라인 > id > class, 속성, pseudo-class > 요소, pseudo-element

3. CSS 파일 로딩 순서

- **CSS 상속**

: CSS는 상속을 통해 부모 요소의 속성을 자식에게 상속한다.

- 속성(프로퍼티) 중에는 상속이 되는 것과 되지 않는 것들 것 있다.

- 상속되는 것 예시

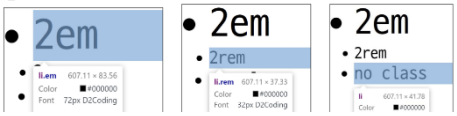
  : Text 관련 요소(font, color, text – align), opacity, visibility

- 상속되지 않는 것 예시

  : Box model 관련 요소(width, height, margin, padding, border, box-sizing, display), position 관련 요소(position, top/right/bottom/left, z-index) 등

[CSS 기본 스타일]

- **크기 단위**



1. **px(픽셀)**

- 모니터 해상도의 한 화소인 '픽셀'기준

- 픽셀의 크기는 변하지 않기 때문에 고정적인 단위

2. **%**

- 백분율 단위

- 가변적인 레이아웃에서 자주 사용

3. **em**

- (바로 위, 부모 요소에 대한) 상속의 영향을 받음

- 배수 단위, 요소에 지정된 사이즈에 상대적인 사이즈를 가짐

4. rem

- (바로 위, 부모 요소에 대한) 상속의 영향을 받지 않음
- 최상위 요소(html)의 사이즈를 기준으로 배수 단위를 가짐

5. 크기 단위**(viewport)**

- 웹 페이지를 방문한 유저에게 바로 보이게 되는 웹 컨텐츠의 영역 (디바이스 화면)
- 디바이스의 viewport를 기준으로 상대적인 사이즈가 결정됨
- **vw, vh, vmin, vmax**

```
<body>
  <h1 class="px">px사용</h1>
  <h1 class="vw">vw사용</h1>
</body>
```

```
<style>
  h1 {
    color: black;
    background-color: pink;
  }
  .px {
    width: 200px;
  }

  .vw {
    width: 50vw;
  }
</style>
```

- px는 브라우저의 크기를 변경해도 그대로
- vw는 브라우저의 크기에 따라 크기가 변함

- **색상 단위**

1. **색상 키워드**(background-color: red;)

- 대소문자를 구분하지 않음
- red, blue, black과 같은 특정 색을 직접 글자로 나타냄

2. **RGB 색상**(background-color: rgb(0, 1255, 0) ;)

: 16진수 표기법 혹은 함수형 표기법을 사용해서 특정 색을 표현하는 방식

- '#' + 16진수 표기법
- rgb() 함수형 표기법

3. **HSL 색상**(background-color: hsl(0, 100%, 50%) ;)

: 색상, 채도, 명도를 통해 특정 색을 표현하는 방식

- 색상, 채도, 명도

4. a는 alpha(투명도)

- **CSS 문서 표현**

- 텍스트

: 서체(font-family), 서체 스타일(font-style, font-weight 등)

: 자간(letter-spacing), 단어 간격(word-spacing),

행간(line-height) 등

- 컬러(color), 배경(background-image, background-color)
- 기타 HTML 태그별 스타일링

: 목록(li), 표(table)

[Selectors 심화]

- **결합자 (Combinators)**

- **자손 결합자 (공백)**

: selectorA 하위의 모든 selectorB 요소

- **자식 결합자 (>)**

: selectorA 바로 아래의 selectorB 요소

- **일반 형제 결합자 (~)**

: selectorA의 형제 요소 중 뒤에 위치하는 selectorB 요소를 모두 선택

- **인접 형제 결합자 (+)**

: selectorA의 형제 요소 중 바로 뒤에 위치하는 selectorB 요소를 선택

• 자손 결합자

• selectorA 하위의 모든 selectorB 요소

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

```
<div>
  <span>이건 빨강입니다.</span>
  <p>이건 빨강이 아닙니다.</p>
  <p>
    <span>이건 빨강입니다.</span>
  </p>
</div>
```

• 자식 결합자

• selectorA 바로 아래의 selectorB 요소

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

```
<div>
  <span>이건 빨강입니다.</span>
  <p>이건 빨강이 아닙니다.</p>
  <p>
    <span>이건 빨강이 아닙니다.</span>
  </p>
</div>
```

• 일반 형제 결합자

• selectorA의 형제 요소 중 뒤에 위치하는 selectorB 요소를 모두 선택

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

```
<span>p태그의 앞에 있기 때문에 이건 빨강이 아닙니다.</span>
<p>여기 문단이 있습니다.</p>
<b>그리고 코드도 있습니다.</b>
<span>p태그와 형제이기 때문에 이건 빨강입니다!</span>
<b>더 많은 코드가 있습니다.</b>
<span>이것도 p태그와 형제이기 때문에 빨강입니다!</span>
```

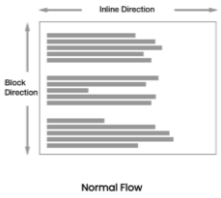
• 인접 형제 결합자

• selectorA의 형제 요소 중 바로 뒤에 위치하는 selectorB 요소를 선택

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

```
<span>p태그의 앞에 있기 때문에 이건 빨강이 아닙니다.</span>
<p>여기 문단이 있습니다.</p>
<span>p태그와 인접한 형제이기 때문에 이건 빨강입니다!</span>
<b>더 많은 코드가 있습니다.</b>
<span>p태그와 인접한 형제가 아니기 때문에 이건 빨강이 아닙니다.</span>
```

[CSS Box model]



모든 요소는 네모(박스모델)이고,

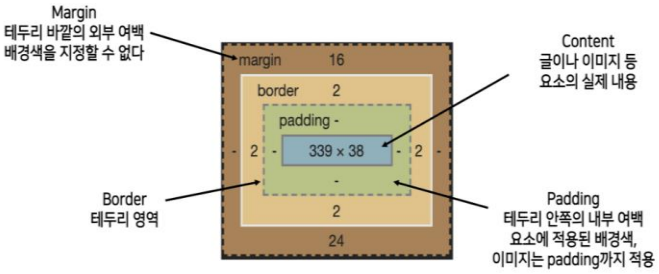
위에서부터 아래로, 왼쪽에서 오른쪽으로 쌓인다.

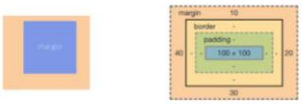
(좌측 상단에 배치)

- **Box model**

- 모든 HTML 요소는 box 형태로 되어있음
- 하나의 박스는 네 부분(영역)으로 이루어짐

1. margin
2. border
3. padding
4. content

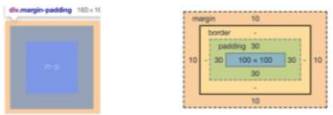




```
.margin {
 margin-top: 10px;
 margin-right: 20px;
 margin-bottom: 30px;
 margin-left: 40px;
}
```

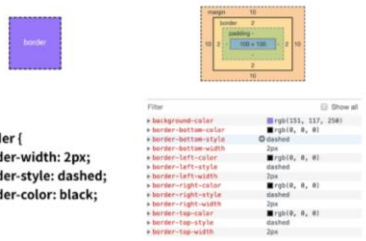
상하좌우!

<margin>



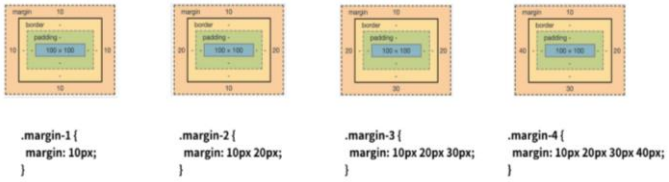
```
.margin-padding {
 margin: 10px;
 padding: 30px;
}
```

<padding>



```
.border {
 border-width: 2px;
 border-style: dashed;
 border-color: black;
}
```

<border>



```
.margin-1 {
 margin: 10px;
}
.margin-2 {
 margin: 10px 20px;
}
.margin-3 {
 margin: 10px 20px 30px;
}
.margin-4 {
 margin: 10px 20px 30px 40px;
}
```

shorthand를 통해서 표현 가능하다.

```
.border {
 border-width: 2px;
 border-style: dashed;
 border-color: black;
}
```

```
.border {
 border: 2px dashed black;
}
```

border도 shorthand가 있다!

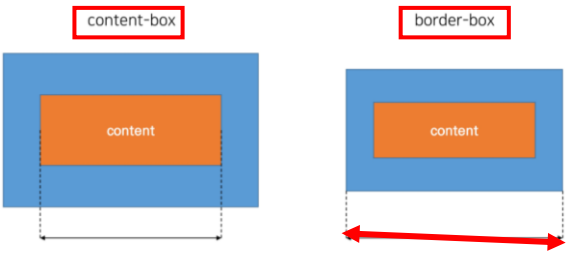
- **box-sizing**

- 기본적으로 모든 요소의 box-sizing은 content-box

  : padding을 제외한 순수 contents 영역만을 box로 지정

- 다만, 우리가 일반적으로 영역을 볼 때는 border까지의 너비를 100px 보는 것을 원함

  : 그 경우 **box-sizing을 border-box로 설정**



[CSS Display]

- **CSS 원칙**

- 모든 요소는 네모(박스모델)이고, 좌측상단에 배치.
- Display에 따라 크기와 배치가 달라진다.

- 대표적으로 활용되는 display

1. Display: **block**

- 줄 바꿈이 일어나는 요소
- 화면 크기 전체의 가로 폭을 차지한다.
- 블록 레벨 요소 안에 인라인 레벨 요소가 들어갈 수 있음

2. Display: **inline**

- **줄 바꿈이 일어나지 않는 행의 일부 요소**
- Content 너비만큼 가로 폭을 차지한다.
- Width, height, margin-top, margin-bottom을 지정할 수 없다.
- 상하 여백은 line-height로 지정한다.

- **블록 레벨 요소와 인라인 레벨 요소**

- 블록 레벨 요소

  : **div / ul, ol, li / p / hr / form**

- 인라인 레벨 요소

  : **span / a / img / input, label / b, em, i, strong**

- Display

1. Display: **inline-block**

- Block과 inline 레벨 요소의 특징을 모두 가짐
- inline처럼 한 줄에 표시할 수 있고, block처럼 width, height, margin 속성을 모두 지정할 수 있음

2. display: **none**

- 해당 요소를 화면에 표시하지 않고, 공간조차 부여되지 않음
- 이와 비슷한 **visibility: hidden**은 해당 요소가 공간은 차지하나 화면에 표시만 하지 않는다.

[CSS position]

: 문서 상에서 요소의 위치를 지정

- **Static**: 모든 태그의 기본 값(기준 위치)

  : 일반적인 요소의 배치 순서에 다름(좌측 상단)

  : 부모 요소 내에 배치될 때는 부모 요소의 위치를 기준으로 배치 됨

- 아래는 좌표 프로퍼티(top, bottom, left, right)를 사용하여 이동 가능

1. **Relative** : 상대 위치

- 자기 자신의 static 위치를 기준으로 이동 (normal flow 유지)
- 레이아웃에서 요소가 차지하는 공간은 static일 때와 같음 (normal position 대비 offset)

2. **Absolute**: 절대 위치

- 요소를 일반적인 문서 흐름에서 제거 후 레이아웃에 공간을 차지하지 않음 (normal flow에서 벗어남)
- Static이 아닌 가장 가까이 있는 부모/조상 요소를 기준으로 이동 (없는 경우 브라우저 화면 기준으로 이동)

3. **Fiexd:** 고정 위치

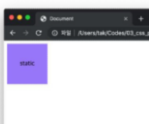
- 요소를 일반적인 문서 흐름에서 제거 후 레이아웃에 공간을 차지하지 않음 (normal flow에서 벗어남)
- 부모 요소와 관계없이 viewport를 기준으로 이동

  : 스크롤 시에도 항상 같은 곳에 위치함

4. **Sticky:** 스크롤에 따라 static -> fixed로 변경

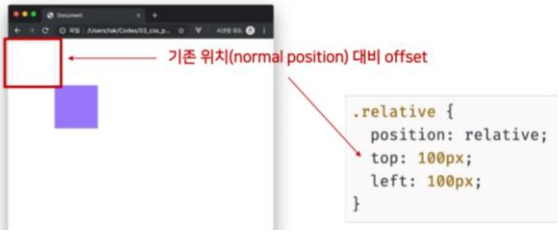
- 속성을 적용한 박스는 평소에 문서 안에서 position: static 상태와 같이 일반적인 흐름에 따르지만 스크롤 위치가 임계점에 이르면 position: fixed와 같이 박스를 화면에 고정할 수 있는 속성
-

static



```
div {
  height: 100px;
  width: 100px;
  background-color: #9775fa;
  color: black;
  line-height: 100px;
  text-align: center;
}
```

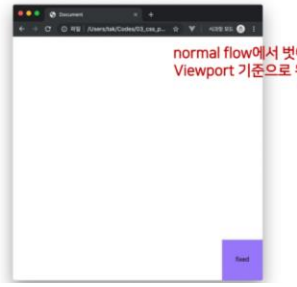
relative



absolute

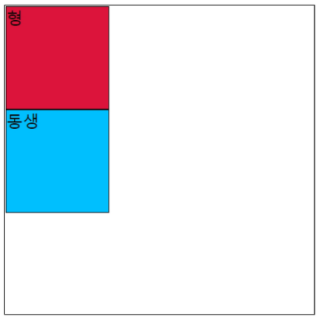

```css
.parent {
  position: relative;
}

.absolute-child {
  position: absolute;
  top: 50px;
  left: 50px;
}
```

fixed




```css
.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
}
```

absolute vs relative



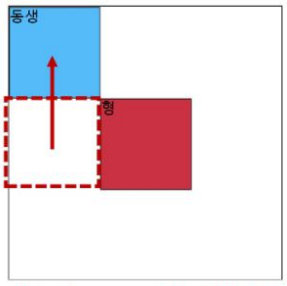

```html
<body>
  <div class="parent">
    <div class="absolute">형</div>
    <div class="sibling">동생</div>
  </div>
  <div class="parent">
    <div class="relative">형</div>
    <div class="sibling">동생</div>
  </div>
</body>
```

```html
<style>
  /* 공통 스타일링 */
  div {
    box-sizing: border-box;
    width: 100px;
    height: 100px;
    border: 1px solid black;
  }

  .parent {
    position: relative;
    width: 300px;
    height: 300px;
  }
</style>
```

```html
<style>
  /* 차이점 확인해보기 */
  .absolute {
    position: absolute;
    top: 100px;
    left: 100px;
    background-color: crimson;
  }

  .sibling {
    background-color: deepskyblue;
  }

  .relative {
    position: relative;
    top: 100px;
    left: 100px;
    background-color: crimson;
  }
</style>
```

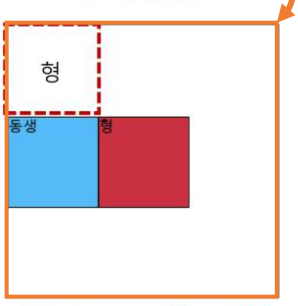
형에게 top: 100px;을 적용했을 때 absolute와 relative의 차이를 알아보자

absolute



absolute는 normal flow에서 벗어남.
즉 다음 블록 요소가 좌측 상단으로 붙음.

relative



relative는 normal flow 유지.
실제 위치는 그대로, 사람 눈에만 이동

<CSS 선택자>

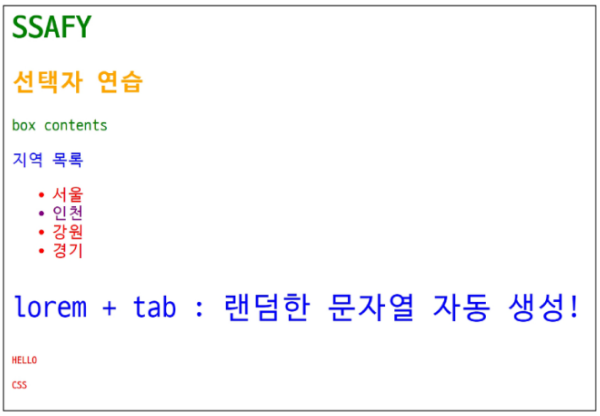
```html
<body>
  <h1 class="green">SSAFY</h1>
  <h2>선택자 연습</h2>
  <div class="green box">
    box contents
    <div>
      <p>지역 목록</p>
      <ul>
        <li>서울</li>
        <li id="purple">인천</li>
        <li>강원</li>
        <li>경기</li>
      </ul>
    </div>
    <p>lorem + tab : 랜덤한 문자열 자동 생성!</p>
  </div>
  <h3>HELLO</h3>
  <h4>CSS</h4>
</body>
```

자손

자식

```css
<style>
  /* 전체 선택자*/
  * {
    color: red;
  }

  /* 요소 선택자 */
  h2 {
    color: orange;
  }

  h3,
  h4 {
    font-size: 10px;
  }
</style>
```

```css
<style>
  /* 클래스 선택자 */
  .green {
    color: green;
  }

  /* id 선택자 */
  #purple {
    color: purple;
  }

  /* 자식 결합자 */
  .box > p {
    font-size: 30px;
  }

  /* 자손 결합자 */
  .box p {
    color: blue;
  }
</style>
```



- 1~8은 무슨 색일까요?

```html
<p>1</p>
<p class="blue">2</p>
<p class="blue green">3</p>
<p class="green blue">4</p>
<p id="red" class="blue">5</p>
<h2 id="red" class="blue">6</h2>
<p id="red" class="blue" style="color: yellow;">7</p>
<h2 id="red" class="blue" style="color: yello;">8</h2>
```

```css
h2 {
  color: darkviolet !important;
}

p {
  color: orange;
}

.blue {
  color: blue;
}

.green {
  color: green;
}

#red {
  color: red;
}
```

1
2
3
4
5
6
7
8