

# 멀티캠퍼스 백엔드 과정 13회차

*K-Digital Training*

강사 임태종

# 수업 내용 인덱스

## 1. CSS 기본 구문

1. CSS 단위
2. 스타일 시트의 작성 방법
3. 선택자와 선언부의 구조
4. 유용한 속성들

## 2.DOM Tree Object

## 3. 스타일 시트의 종류

1. 내부 스타일 시트
2. 외부 스타일 시트
3. 인라인 스타일

## 4. 상속과 우선순위

1. 상속의 개념과 동작 방식
2. 우선순위 규칙과 적용 우선순위

## 5. 박스 모델

1. 박스 모델의 개념과 구성 요소
2. 박스 모델의 속성과 값

## 6. 레이아웃

1. 플로팅과 클리어링
2. 포지셔닝
3. 플렉스 박스 레이아웃

# CSS 기본 구문

```
선택자 {  
    속성: 값;  
    속성: 값;  
    속성: 값;  
}
```

- CSS는 선택자와 선언부로 이루어져 있습니다.
- 선택자는 스타일을 적용할 HTML 요소를 선택하는 부분입니다.
- 선언부는 선택된 HTML 요소에 적용될 스타일 속성과 값의 쌍으로 이루어진 부분입니다.

# CSS 기본 구문

```
선택자 {  
    속성: 값;  
    속성: 값;  
    속성: 값;  
}
```

- 위의 코드에서 선택자는 스타일을 적용할 HTML 요소를 선택하는 부분입니다. 속성은 HTML 요소에 적용할 스타일 속성을 나타내며, 값은 해당 스타일 속성에 적용할 값입니다.
- 예를 들어, p 태그에 대해 글자 색을 빨간색으로 지정하고, 글자 크기를 18px로 지정하고 싶다면 다음과 같이 작성할 수 있습니다

```
p {  
    color: red;  
    font-size: 18px;  
}
```

# CSS 기본 구문

```
p {  
  color: red;  
  font-size: 18px;  
}
```

- 위의 코드에서 p는 p 태그를 선택하는 선택자이고, color와 font-size는 스타일 속성이며, red와 18px는 해당 스타일 속성에 적용할 값입니다.

# CSS 심화 구문

- 아래 세 가지를 좀 더 깊게 순서대로 알아보겠습니다.
  - CSS 주석
  - 박스모델
  - CSS 단위

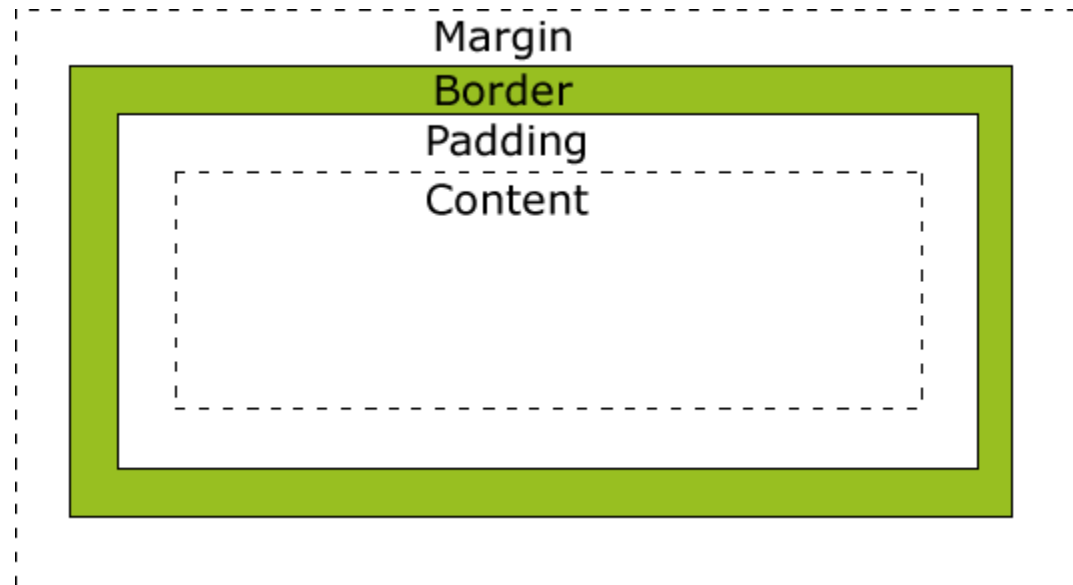
# CSS 주석

- CSS 주석은 /\*와 \*/로 감싸진 부분으로, 코드에 대한 설명이나 비활성화를 위해 사용됩니다.
- 주석으로 된 부분은 사람을 위한 부분이며 기계는 이 부분을 무시합니다.
- 주석을 이용해서 문서를 관리하는 방법은 다양하게 있으며 이 부분은 자바 과정 진행하면서 후술하겠습니다.

```
/* 이 부분은 CSS 주석입니다. */
```

# 박스 모델

- 박스 모델은 HTML 요소를 박스 형태로 간주하여, 그 안에 들어갈 수 있는 내용과 크기, 여백 등을 정의합니다.
- 아래의 그림에서 content는 HTML 요소 내용을 나타내며, padding은 요소 내용과 테두리(border) 사이의 여백을 나타냅니다. border는 요소 주변의 경계선을 나타내며, margin은 요소와 요소 사이의 여백을 나타냅니다.
- 박스 모델과 관련된 CSS 속성으로는 width, height, padding, border, margin 등이 있습니다





# 박스 모델

- 아래의 코드는 .box 클래스를 가진 HTML 요소에 대해, 너비가 200px, 높이가 100px, 안쪽 여백(padding)이 20px, 경계선(border)이 1px 두께의 검은색(solid black), 바깥쪽 여백(margin)이 10px인 박스 모델을 적용합니다.



```
.box {  
  width: 200px;  
  height: 100px;  
  padding: 20px;  
  border: 1px solid black;  
  margin: 10px;  
}
```

# CSS 단위

- CSS는 크기, 길이, 각도, 색상 등 다양한 값의 단위를 지원합니다.
- 대표적인 단위로는 px, em, rem, %, vh, vw 등이 있습니다.
  - px: 픽셀 단위
  - em: 상위 요소의 글꼴 크기를 기준으로 하는 상대적인 크기 단위
  - rem: 최상위 요소(보통 html 태그)의 글꼴 크기를 기준으로 하는 상대적인 크기 단위
  - %: 상위 요소의 크기에 대한 비율을 나타내는 단위
  - vh: 뷰포트 세로 길이의 1/100
  - vw: 뷰포트 가로 길이의 1/100

```
.box {  
  font-size: 16px;  
  width: 50%;  
  padding: 1em;  
  margin: 2rem;  
  height: 50vh;  
}
```

# CSS 단위

- CSS는 크기, 길이, 각도, 색상 등 다양한 값의 단위를 지원합니다. 다양한 단위를 사용하는 이유는 웹 페이지의 다양한 크기와 레이아웃에 적합한 값을 지정하기 위해서입니다.
- 단위의 종류는 아래와 같습니다.
  - 길이 단위
  - 색상 단위
  - 미디어 쿼리 단위
  - ~~각도 단위~~ (프론트엔드 과정이 아니므로 아곳에선 생략)
  - ~~시간 단위~~ (프론트엔드 과정이 아니므로 아곳에선 생략)
- CSS 함수는 여기서는 자세히 다루지 않으며 PPT 자료로만 남기겠습니다.

# CSS 단위 - 길이 단위 (px)

- px
  - px는 픽셀(pixel) 단위로, 가장 많이 사용되는 길이 단위입니다. 픽셀은 화면에 표시되는 하나의 점을 나타내는 단위이며, 고정적인 크기를 지정할 때 주로 사용됩니다.

```
.box {  
  width: 200px;  
  height: 100px;  
}
```

# CSS 단위 - 길이 단위 (em)

- em
  - em은 요소의 글꼴 크기를 기준으로 하는 상대적인 길이 단위입니다. 부모 요소의 글꼴 크기에 따라 자식 요소의 크기도 함께 변화합니다.
  - 코드에서 "h1" 요소의 글꼴 크기는 2em으로 지정되어 있으므로, 기본 글꼴 크기의 2배로 설정됩니다. "p" 요소의 글꼴 크기는 1.2em으로 지정되어 있으므로, 기본 글꼴 크기의 1.2배로 설정됩니다.
  - 예를 들어, 기본 글꼴 크기가 16px인 경우, "h1" 요소의 글꼴 크기는 32px이 되고 "p" 요소의 글꼴 크기는 19.2px이 됩니다. 이렇게 em 단위를 사용하면 상대적인 글꼴 크기를 쉽게 지정할 수 있습니다.

```
<p>기본글꼴 사이즈 입니다.</p>  
<p style="font-size: 2em;">Hello, World!</p>  
<p style="font-size: 1.2em;">This is an example of using em units in CSS.</p>
```

# CSS 단위 - 길이 단위 (em)

```
<html>
  <head>
    <style>
      body {
        font-size: 16px; /* 기본 폰트 크기 설정 */
      }
      .wrapper {
        width: 30em; /* em 단위 사용 */
        margin: 0 auto; /* 가운데 정렬 */
        background-color: #f2f2f2; /* 배경색 설정 */
        padding: 1em; /* 여백 설정 */
      }
      h1 {
        font-size: 2em; /* 부모 요소의 폰트 크기의 2배 */
        margin-bottom: 0.5em; /* 아래쪽 여백 */
      }
      p {
        font-size: 1em; /* 부모 요소의 폰트 크기와 같게 설정 */
        margin-bottom: 1em; /* 아래쪽 여백 */
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
      <h1>Em Example</h1>
      <p>CSS에서 em 단위를 사용하여 유연한 레이아웃을 만드는 예입니다.</p>
      <p>em 단위를 사용하여 사용자가 선호하는 글꼴 크기에 맞게 조정되는 레이아웃을 만들 수 있습니다.</p>
    </div>
  </body>
</html>
```

# CSS 단위 - 길이 단위 (rem)

- rem
  - rem은 em과 비슷하지만, 상위 요소의 폰트 크기에 대한 상대적인 크기가 아니라 루트 요소 (HTML 요소)의 폰트 크기에 대한 상대적인 크기를 나타냅니다.
  - 예를 들어, HTML 요소의 폰트 크기가 16px인 경우, 다음과 같은 코드를 사용하여 폰트 크기를 지정할 수 있습니다.

예제코드 다음페이지에

```
<html>

<head>
  <style>
    html {
      font-size: 20px;
      /* 기본 폰트 크기 설정 */
    }

    .container {
      font-size: 1.2rem;
      /* 루트 요소 폰트 크기의 1.2배 */
    }

    h1 {
      font-size: 2em;
      /* 부모 요소의 폰트 크기의 2배 */
    }

    p {
      font-size: 1.5em;
      /* 부모 요소의 폰트 크기의 1.5배 */
      margin-bottom: 1rem;
      /* 루트 요소 폰트 크기의 1배 */
    }
  </style>
</head>

<body>
  <div class="container">
    <h1>Em and Rem Example</h1>
    <p>CSS에서 em과 rem 단위를 함께 사용하는 예입니다.</p>
    <p>em 및 rem 단위를 사용하여 글꼴 크기를 더 유연하게 만들고 상위 요소에 상대적으로 만들 수 있습니다.</p>
  </div>
</body>

</html>
```



# CSS 단위 - 길이 단위 (rem)

- 이전 코드에서, "body" 요소의 폰트 크기를 16px로 설정하고, "container" 클래스를 가진 요소의 폰트 크기를 루트 요소 폰트 크기의 1.2배로 설정합니다. "h1" 요소는 부모 요소인 "container"의 폰트 크기의 2배로 설정되고, "p" 요소는 부모 요소인 "container"의 폰트 크기의 1.5배로 설정됩니다.
- 또한, "p" 요소에는 "margin-bottom" 속성이 있으며, 값으로 "1rem"을 지정했습니다. 이 경우, "p" 요소의 아래쪽 여백이 루트 요소 폰트 크기의 1배로 설정됩니다.
- 이렇게 em과 rem을 함께 사용하면, 상대적인 글꼴 크기와 여백을 조절하여 보다 유연하게 레이아웃을 구성할 수 있습니다.

# CSS 단위 - 길이 단위 (%)

- %
  - "%"는 상대적인 단위 중 하나로, 부모 요소의 크기에 대한 백분율을 나타냅니다.
  - 다음페이지에는 "%" 단위를 사용하여 레이아웃을 구성하는 예제가 있습니다.
  - 다음 페이지에 있는 코드해석입니다.
  - "wrapper" 클래스를 가진 요소의 너비를 부모 요소의 너비의 80%로 설정합니다.
  - 이후, "box" 클래스를 가진 요소의 너비를 부모 요소의 너비의 50%로 설정하고, 높이를 100px로 설정합니다.
  - "box" 클래스를 가진 요소는 위아래 여백이 20px이며, 가운데 정렬이 됩니다.
  - 이렇게 함으로써, "wrapper" 요소의 너비에 따라 "box" 요소의 크기가 유동적으로 조절되면서 레이아웃을 구성할 수 있습니다.
  - 예제에서는 "%" 단위를 주로 너비와 관련된 속성에 사용하였으나, 다양한 속성에 사용이 가능합니다.

# CSS 단위 - 길이 단위 (%)

```
<html>
  <head>
    <style>
      .wrapper {
        width: 80%; /* 부모 요소의 너비의 80% */
        margin: 0 auto; /* 가운데 정렬 */
        background-color: #f2f2f2; /* 배경색 설정 */
      }
      .box {
        width: 50%; /* 부모 요소의 너비의 50% */
        height: 100px; /* 높이 설정 */
        margin: 20px auto; /* 위아래 여백은 20px, 가운데 정렬 */
        background-color: #3498db; /* 배경색 설정 */
        color: #fff; /* 글자색 설정 */
        text-align: center; /* 가운데 정렬 */
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
      <div class="box">Box 1</div>
      <div class="box">Box 2</div>
    </div>
  </body>
</html>
```

# CSS 단위 - 길이 단위 (vw, vh)

- 먼저, vw 단위는 뷰포트의 너비에 대한 상대적인 크기를 나타내며, vh 단위는 뷰포트의 높이에 대한 상대적인 크기를 나타냅니다.
- 뷰포트(viewport)란, 사용자가 웹사이트를 보는 디바이스의 화면 영역을 말합니다. 뷰포트는 웹사이트를 렌더링할 때 중요한 역할을 합니다. 뷰포트의 크기에 따라 웹사이트가 어떻게 표시되는지 결정됩니다.
- 모바일 디바이스의 경우, 뷰포트 크기가 작기 때문에 웹사이트의 레이아웃이 바뀌어야 합니다. 이를 위해 viewport meta 태그를 사용하여 뷰포트의 크기를 조절할 수 있습니다.
- 아래의 코드는 뷰포트의 너비를 디바이스의 너비로 설정하고, 초기 확대/축소 비율을 1로 설정하는 예제입니다.
- 이렇게 하면 디바이스의 뷰포트 크기에 맞게 웹사이트를 보여줄 수 있습니다.

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

# CSS 단위 - 길이 단위 (vw, vh)

```
<html>
<head>
  <style>
    .box {
      width: 50vw;
      /* 뷰포트의 너비의 50% */
      height: 50vh;
      /* 뷰포트의 높이의 50% */
      background-color: #3498db;
      /* 배경색 설정 */
      color: #fff;
      /* 글자색 설정 */
      text-align: center;
      /* 가운데 정렬 */
    }
  </style>
</head>
<body>
  <div class="box">50% Width and Height of Viewport</div>
</body>
</html>
```

- 위의 코드에서, "box" 클래스를 가진 요소의 너비는 뷰포트의 너비의 50%로, 높이는 뷰포트의 높이의 50%로 설정됩니다.
- 이렇게 함으로써, 뷰포트의 크기에 상관없이 항상 50%의 크기를 가지는 사각형을 생성할 수 있습니다.

# CSS 단위 - 길이 단위 (vw, vh)

- 뷰포트 크기에 따라 폰트 크기를 조절할 수도 있습니다.
- "body" 요소의 폰트 크기는 뷰포트의 너비의 1.5%로 설정되고, "h1" 요소의 폰트 크기는 뷰포트의 높이의 3%로 설정됩니다. 이렇게 함으로써, 뷰포트의 크기에 따라 폰트 크기가 조절되어 반응형 레이아웃을 생성할 수 있습니다.

```
<html>
  <head>
    <style>
      body {
        font-size: 1.5vw; /* 뷰포트 너비의 1.5% */
      }
      h1 {
        font-size: 3vh; /* 뷰포트 높이의 3% */
        text-align: center;
      }
    </style>
  </head>
  <body>
    <h1>뷰포트 단위 예시</h1>
    <p>CSS에서 뷰포트 단위를 사용하여 반응형 레이아웃을 만드는 예입니다.</p>
  </body>
</html>
```

# CSS 단위 - 색깔 단위

- CSS에서는 다양한 색상 단위를 제공합니다. 가장 기본적인 것은 이름으로 색상을 지정하는 것입니다. 예를 들어, "red"는 빨간색을 의미합니다.

```
<html>
  <head>
    <style>
      body {
        background-color: lightgray; /* 배경색 설정 */
      }
      h1 {
        color: red; /* 글자색 설정 */
      }
    </style>
  </head>
  <body>
    <h1>Color Units Example</h1>
    <p>This is an example of using named color units in CSS.</p>
  </body>
</html>
```

# CSS 단위 - 색깔 단위

- 위의 코드에서, "box" 클래스를 가진 요소의 너비와 높이를 100px로 설정하고, 배경색을 RGBa 색상 단위를 사용하여 설정했습니다.
- "border" 속성은 HEX 색상 단위를 사용하여 설정하고, "box-shadow" 속성은 HSL 색상 단위를 사용하여 설정했습니다.
- 각각의 색상 단위는 다양한 방식으로 색상을 지정할 수 있습니다.
- 예를 들어, RGB 단위는 빨강(Red), 녹색(Green), 파랑(Blue) 값의 조합으로 색상을 표현하며, HEX 단위는 16진수 값으로 색상을 표현합니다.
- "red", "yellow" 같은 색의 정보는 표준으로 정해져 있으며 아래 링크에서 더 많은 정보를 확인 할 수 있습니다.
  - <https://www.w3.org/TR/css-color-4/#named-colors>
  - W3C에서는 CSS의 색상 이름과 해당 색상의 RGB 값, 색상 이름의 예시 등을 제공합니다. 이 페이지에서는 "lightgray"뿐만 아니라, 다양한 이름의 색상 정보를 확인할 수 있습니다.



# CSS 단위 - 미디어 쿼리

- 미디어 쿼리(Media Query)는 CSS에서 사용되는 반응형 디자인 기술 중 하나로, 특정 미디어(화면, 프린트 등)의 크기나 방향 등을 검사하여 스타일을 변경하는 것입니다. 이를 통해, 다양한 미디어에서 최적화된 디자인을 제공할 수 있습니다.
- 아래는 다음페이지에 있는 미디어 쿼리 예제의 해설입니다.
  - "body" 요소의 기본 배경색은 "white"로 설정되어 있습니다. "@media" 블록을 사용하여 미디어 쿼리를 정의하고, 화면 크기에 따라 배경색을 변경합니다.  
"@media (max-width: 768px)"는 화면의 최대 너비가 768px 이하인 경우에 적용되며, "@media (min-width: 769px)"는 화면의 최소 너비가 769px 이상인 경우에 적용됩니다. 각각의 경우에 따라 배경색이 "lightblue"와 "lightgreen"으로 변경됩니다.

예제코드 다음페이지에

# CSS 단위 - 미디어 쿼리

```
<html>

<head>
  <style>
    body {
      background-color: white;
    }

    @media (max-width: 768px) {

      /* 768px 이하인 경우 */
      body {
        background-color: lightblue;
      }
    }

    @media (min-width: 769px) {

      /* 769px 이상인 경우 */
      body {
        background-color: lightgreen;
      }
    }
  </style>
</head>
<body>
  <h1>Media Query Example</h1>
  <p>CSS에서 미디어 쿼리를 사용한 예입니다.</p>
</body>
</html>
```

# CSS 단위 - 미디어 쿼리

- 미디어 쿼리에서 "screen"은 화면을 가지고 있는 디바이스를 의미합니다.
- 화면을 가지지 않는 디바이스에는 예를 들면 음성 인식 기기, 스마트 스피커, 프린터 등이 있습니다.
- 그렇기 때문에, 이러한 화면을 가지지 않는 디바이스에서는 "screen" 대신 "print" 또는 "speech"와 같은 미디어 타입을 사용하여 스타일을 적용합니다.
- 예를 들어, 프린터에서 출력될 때 스타일을 적용하고 싶다면, "@media print" 미디어 쿼리를 사용할 수 있습니다.
- 통상적인 사이즈는 아래와 같습니다.
  - 320px — 480px: 모바일 기기
  - 481px — 768px: 태블릿 (아이패드, 갤럭시탭)
  - 769px — 1024px: 노트북, 작은 사이즈 모니터
  - 1025px — 1200px: 데스크탑, 큰 사이즈 모니터
  - 1201px and more — TV

# CSS 단위 - 미디어 쿼리

- 미디어 쿼리 - screen

```
<html>

<head>
  <style>
    /* 기본 스타일 */
    body {
      font-size: 16px;
      color: black;
    }

    /* 미디어 쿼리 */
    @media only screen and (max-width: 767px) {
      body {
        background-color: gray;
        font-size: 14px;
        color: gray;
      }
    }
  </style>
</head>

<body>
  <h1>Media Query Example</h1>
  <p>CSS에서 미디어 쿼리를 사용한 예입니다.</p>
</body>

</html>
```

# CSS 단위 - 미디어 쿼리

- print / speech 등의 예제
  - 프린터에서 출력될 때 스타일을 적용하고 싶다면, "@media print" 미디어 쿼리를 사용할 수 있습니다.
  - 음성 인식 기기에서 스타일을 적용하고 싶다면, "@media speech" 미디어 쿼리를 사용할 수 있습니다.

```
@media print {  
    /* 프린터에서 적용될 스타일 (인쇄페이지용) */  
}
```

```
@media speech {  
    /* 음성 인식 기기에서 적용될 스타일 */  
}
```

# CSS 단위 - 주의사항

- 상대적인 단위 사용 시 주의점
  - 상대적인 단위를 사용할 때에는 부모 요소의 크기나 글꼴 크기 등이 변경될 경우, 자식 요소의 크기도 함께 변경될 수 있습니다. 이를 방지하기 위해서는 rem 과 같이 최상위 요소를 기준으로 하는 상대적인 단위를 사용하는 것이 좋습니다.
- 다양한 기기에 대한 대응
  - 다양한 기기에서 웹 페이지가 잘 보이도록 하기 위해서는 다양한 기기의 화면 크기와 해상도를 고려하여 적절한 단위를 선택해야 합니다. 대표적인 예로는 반응형 웹 디자인을 통해 다양한 기기에 대응할 수 있습니다.

# CSS 단위 - 주의사항

- 단위의 의미 파악
  - 단위는 숫자와 함께 사용되어 그 크기나 의미를 나타내는데, 단위의 의미를 파악하지 못하면 원하는 결과를 얻지 못할 수 있습니다. 따라서 단위의 의미를 잘 파악하고 사용해야 합니다.
- 너무 많은 단위 사용하지 않기
  - CSS에서는 다양한 단위를 지원하지만, 너무 많은 단위를 사용하면 코드의 가독성이 떨어지고 유지보수가 어려워질 수 있습니다. 따라서 필요한 경우에만 적절한 단위를 사용하고, 너무 많은 단위는 피하는 것이 좋습니다.

# 스타일 시트 작성 방법

## 1.CSS 파일 생성

- 우선 CSS 스타일 시트를 작성하기 위해서는 .css 확장자를 가진 새로운 파일을 생성해야 합니다. 파일 이름은 원하는 대로 설정할 수 있습니다.

## 2.HTML 파일과 연결

- CSS 파일을 작성한 후에는, HTML 파일과 연결해야 합니다. 이를 위해서는 HTML 파일의 <head> 태그 안에 <link> 태그를 추가합니다.

```
<head>  
  <link rel="stylesheet" href="css/6_rel.css">  
</head>
```

- 위 코드에서 "href" 속성에는 CSS 파일의 경로를 지정합니다. 이 예시에서는 현재 파일과 같은 디렉토리 내에 있는 "style.css" 파일과 연결하고 있습니다.



# 스타일 시트 작성 방법

## 3. 스타일 작성

- CSS 파일과 HTML 파일을 연결했다면 이제 스타일을 나타낼 수 있습니다. CSS 파일에서는 각각의 요소에 대한 스타일을 정의하며 이를 선택자(selector)와 스타일 규칙(style rule)으로 구성합니다.

```
/* 선택자 */  
h1 {  
  /* 스타일 규칙 */  
  color: blue;  
  font-size: 24px;  
}
```

- 위 코드에서 "h1"은 선택자(selector)로, 이는 HTML 파일에서 "h1" 태그에 해당하는 요소를 선택하여 스타일을 적용하는 역할을 합니다. "{}" 내에 있는 "color"와 "font-size"는 스타일 규칙(style rule)으로, 선택한 요소에 대한 스타일을 지정합니다.

# 스타일 시트 작성 방법

## 4.HTML 파일에서 스타일 적용

- CSS 파일에 스타일을 작성한 후에는, HTML 파일에서 이를 적용합니다. 이를 위해서는 HTML 요소에 "class"나 "id"와 같은 속성을 추가하고, CSS 파일에서 이를 선택하여 스타일을 적용합니다.

```
<html>
<head>
  <link rel="stylesheet" href="css/6_rel.css">
</head>

<body>
  <div>
    <div class="title">
      <p>This is title and p</p>
    </div>
  </div>
  <h1 class="title">This is title class</h1>
</body>

</html>
```

- 위 코드에서 "class" 속성을 사용하여 "title"이라는 클래스를 추가하고 있습니다. 이를 CSS 파일에서 선택하여 스타일을 적용할 수 있습니다.

# 스타일 시트 작성 방법

- 파일 "`css/6_rel.css`" 의 내용

```
/* 선택자 */  
h1 {  
    /* 스타일 규칙 */  
    color: blue;  
    font-size: 10px;  
}  
  
.title {  
    color: blueviolet;  
    font-size: 10px;  
}  
  
.title p {  
    color: blueviolet;  
    font-size: 50px;  
}
```

# 선택자와 선언부의 구조

- CSS에서 선택자(selector)는 HTML 요소를 선택하여 스타일을 적용하는 역할을 합니다. 선택자를 사용하여 HTML 요소를 선택할 수 있는 방법은 다양합니다.
  - 요소 선택자 (Element Selector)
  - 클래스 선택자 (Class Selector)
  - ID 선택자 (ID Selector)
  - 후손 선택자 (Descendant Selector)
  - 자식 선택자 (Child Selector)
  - 인접 형제 선택자 (General Sibling Selector)

# 선택자와 선언부의 구조

- 모든 요소 선택자
  - 모든 요소에 스타일을 적용하기 위해 \* 선택자를 사용할 수 있습니다

```
* {  
  color: blue;  
}
```

- 요소 선택자
  - 요소 선택자는 HTML 요소의 이름을 사용하여 선택합니다. 요소 선택자는 해당 HTML 요소 타입의 모든 요소에 적용됩니다.
  - 예를 들어, 다음 코드에서 "h1" 요소 선택자는 HTML 파일 내의 모든 "h1" 태그에 스타일을 적용합니다.

```
h1 {  
  font-size: 32px;  
}
```

# 선택자와 선언부의 구조

- 클래스 선택자
  - 클래스 선택자는 "." 기호를 사용하여 선택합니다. 클래스는 여러 개의 요소에 적용할 수 있으며, 한 요소에 여러 개의 클래스를 지정할 수도 있습니다.
  - 예를 들어, 다음 코드에서 ".intro" 클래스 선택자는 HTML 파일 내의 "class" 속성 값이 "intro"인 모든 요소에 스타일을 적용합니다.

```
.intro {  
  background-color: yellow;  
}
```

# 선택자와 선언부의 구조

- ID 선택자
  - ID 선택자는 "#" 기호를 사용하여 선택합니다. ID는 한 요소에만 적용할 수 있으며, 중복되지 않아야 합니다.
  - 예를 들어, 다음 코드에서 "#main" ID 선택자는 HTML 파일 내의 "id" 속성 값이 "main"인 요소에 스타일을 적용합니다.

```
#main {  
    font-size: 24px;  
}
```

# 선택자와 선언부의 구조

- 후손 선택자
  - 후손 선택자는 두 개 이상의 요소를 연결하여 선택합니다. 후손 선택자를 사용하여 하위 요소에 스타일을 적용할 수 있습니다.
  - 예를 들어, 다음 코드에서 "body p" 후손 선택자는 HTML 파일 내의 "body" 태그 내에 있는 모든 "p" 태그에 스타일을 적용합니다.

```
body p {  
    color: blue;  
}
```



# 선택자와 선언부의 구조

- 자식 선택자
  - 자식 선택자는 ">" 기호를 사용하여 선택합니다. 자식 요소는 바로 아래 단계에 있는 요소만 선택됩니다.
  - 예를 들어, 다음 코드에서 "div > p" 자식 선택자는 "div" 태그의 자식으로 바로 아래에 있는 "p" 태그만 선택합니다.

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

# 선택자와 선언부의 구조

- 인접 형제 선택자
  - 인접 형제 선택자는 "+" 기호를 사용하여 선택합니다. 선택한 요소와 동일한 부모 요소를 가진, 바로 옆에 있는 형제 요소만 선택됩니다.
  - 예를 들어, 다음 코드에서 "h1 + p" 인접 형제 선택자는 "h1" 태그 바로 다음에 있는 "p" 태그만 선택합니다.

```
h1 + p {  
  font-size: 16px;  
}
```

# 선택자와 선언부의 구조

- 일반 형제 선택자
  - 일반 형제 선택자는 "~" 기호를 사용하여 선택합니다. 선택한 요소와 동일한 부모 요소를 가진, 모든 형제 요소가 선택됩니다.
  - 예를 들어, 다음 코드에서 "h1 ~ p" 일반 형제 선택자는 "h1" 태그 다음에 있는 모든 "p" 태그를 선택합니다.

```
h1 ~ p {  
  color: green;  
}
```

# 선택자와 선언부의 구조

- 속성 선택자(Attribute Selector)
  - 속성 선택자는 HTML 요소의 속성을 선택하여 스타일을 적용합니다. 속성 선택자는 대괄호 "[" 안에 속성 이름과 속성 값으로 구성됩니다.
  - 예를 들어, 다음 코드에서 "[href]" 속성 선택자는 "href" 속성이 있는 모든 요소를 선택합니다.

```
[href] {  
  color: purple;  
}
```

# 선택자와 선언부의 구조

- 속성 값 선택자(Attribute Value Selector)
  - 속성 값 선택자는 속성 값이 일치하는 요소를 선택합니다. "=" 기호를 사용하여 속성 값과 일치하는 값을 지정합니다.
  - 예를 들어, 다음 코드에서 "[href='https://example.com']" 속성 값 선택자는 "href" 속성 값이 "https://example.com"인 요소를 선택합니다.

```
[href='https://example.com'] {  
    font-size: 20px;  
}
```

# 선택자와 선언부의 구조

- 시작 부분 일치 선택자(Beginning Matching Selector)
  - 시작 부분 일치 선택자는 "^" 기호를 사용하여 속성 값의 시작 부분과 일치하는 요소를 선택합니다.
  - 예를 들어, 다음 코드에서 "[class^='para']" 시작 부분 일치 선택자는 "class" 속성 값이 "para"로 시작하는 모든 요소를 선택합니다.

```
[class^='para'] {  
  font-weight: bold;  
}
```

# 선택자와 선언부의 구조

- 종료 부분 일치 선택자(Ending Matching Selector)
  - 종료 부분 일치 선택자는 "\$" 기호를 사용하여 속성 값의 종료 부분과 일치하는 요소를 선택합니다.
  - 예를 들어, 다음 코드에서 "[class\$='para']" 종료 부분 일치 선택자는 "class" 속성 값이 "para"로 끝나는 모든 요소를 선택합니다.

```
[class$='para'] {  
    font-style: italic;  
}
```

# 선택자와 선언부의 구조

- 포함하는 부분 일치 선택자(Substring Matching Selector)
  - 포함하는 부분 일치 선택자는 "\*" 기호를 사용하여 속성 값에 특정 문자열이 포함되어 있는 요소를 선택합니다.
  - 예를 들어, 다음 코드에서 "[class\*='para']" 포함하는 부분 일치 선택자는 "class" 속성 값에 "para" 문자열이 포함된 모든 요소를 선택합니다.

```
[class*='para'] {  
  text-decoration: underline;  
}
```



# 선택자와 선언부의 구조

- 여러 개의 요소를 선택하기 위해 쉼표(,)를 사용할 수 있습니다:

```
h1, h2, h3 {  
  font-family: Arial, sans-serif;  
}
```

# 선택자와 선언부의 구조

- 가상 선택자(pseudo-selector)는 HTML 요소의 특정 상태를 선택하는 데 사용됩니다. 가상 선택자는 콜론(:)으로 시작하며, 선택자 뒤에 추가하여 사용합니다.
- 여기에 몇 가지 가상 선택자에 대한 예시를 들어보겠습니다.
- :hover 가상 선택자
  - :hover 가상 선택자는 요소 위에 마우스 커서가 올라갔을 때 선택됩니다. 이 가상 선택자를 사용하면, 요소에 마우스를 올렸을 때 스타일을 변경할 수 있습니다.
  - 예를 들어, 다음 코드에서 "button:hover" 가상 선택자는 마우스 커서가 "button" 요소 위에 있을 때 스타일을 변경합니다.

```
button:hover {  
    background-color: yellow;  
}
```

# 선택자와 선언부의 구조

## 2.:active 가상 선택자

- :active 가상 선택자는 요소를 활성화할 때 선택됩니다. 이 가상 선택자를 사용하면, 요소를 클릭했을 때 스타일을 변경할 수 있습니다.
- 예를 들어, 다음 코드에서 "button:active" 가상 선택자는 "button" 요소를 클릭할 때 스타일을 변경합니다.

```
button:active {  
    background-color: green;  
}
```

# 선택자와 선언부의 구조

- :focus 가상 선택자
  - :focus 가상 선택자는 요소가 포커스를 받았을 때 선택됩니다. 이 가상 선택자를 사용하면, 요소에 포커스가 맞출 때 스타일을 변경할 수 있습니다.
  - 예를 들어, 다음 코드에서 "input:focus" 가상 선택자는 "input" 요소에 포커스가 맞출 때 스타일을 변경합니다.

```
button:focus {  
    outline: none;  
    box-shadow: 0 0 5px blue;  
}
```

# 선택자와 선언부의 구조

- ::before 가상 선택자
  - ::before 가상 선택자는 요소의 내용 앞에 콘텐츠를 추가합니다. 이 가상 선택자를 사용하면, 요소의 내용 앞에 특정 콘텐츠를 추가하여 스타일을 변경할 수 있습니다.
  - 예를 들어, 다음 코드에서 ".box::before" 가상 선택자는 ".box" 요소의 내용 앞에 특정 콘텐츠를 추가합니다.

```
.box::before {  
  content: "→";  
  color: red;  
}
```

# 선택자와 선언부의 구조

- ::after 가상 선택자
  - ::after 가상 선택자는 요소의 내용 뒤에 콘텐츠를 추가합니다. 이 가상 선택자를 사용하면, 요소의 내용 뒤에 특정 콘텐츠를 추가하여 스타일을 변경할 수 있습니다.
  - 예를 들어, 다음 코드에서 ".box::after" 가상 선택자는 ".box" 요소의 내용 뒤에 특정 콘텐츠를 추가합니다.

```
.box::after {  
  content: "<<";  
  color: red;  
}
```

# 선택자와 선언부의 구조

- :first-child 가상 선택자
  - :first-child 가상 선택자는 해당 요소가 부모 요소의 첫 번째 자식 요소일 때 선택됩니다. 이 가상 선택자를 사용하면, 첫 번째 자식 요소에 대해서만 스타일을 적용할 수 있습니다.
  - 예를 들어, 다음 코드에서 "li:first-child" 가상 선택자는 "li" 요소 중 부모 요소의 첫 번째 자식 요소인 요소에 대해서만 스타일을 적용합니다

```
li:first-child {  
    font-weight: bold;  
}
```

# 선택자와 선언부의 구조

- :last-child 가상 선택자
  - :last-child 가상 선택자는 해당 요소가 부모 요소의 마지막 자식 요소일 때 선택됩니다. 이 가상 선택자를 사용하면, 마지막 자식 요소에 대해서만 스타일을 적용할 수 있습니다.
  - 예를 들어, 다음 코드에서 "li:last-child" 가상 선택자는 "li" 요소 중 부모 요소의 마지막 자식 요소인 요소에 대해서만 스타일을 적용합니다.

```
li:first-child {  
    font-weight: bold;  
}
```



# 선택자와 선언부의 구조

- :nth-child(n) 가상 선택자
  - :nth-child(n) 가상 선택자는 해당 요소가 부모 요소의 n번째 자식 요소일 때 선택됩니다. 이 가상 선택자를 사용하면, 특정 순서에 해당하는 자식 요소에 대해서만 스타일을 적용할 수 있습니다.
  - 예를 들어, 다음 코드에서 "li:nth-child(2)" 가상 선택자는 "li" 요소 중 부모 요소의 두 번째 자식 요소인 요소에 대해서만 스타일을 적용합니다.

```
li:nth-child(2) {  
    background-color: blue;  
}
```

# 선택자와 선언부의 구조

- 모든 요소에 파란색 글씨체를 적용하기

```
<!DOCTYPE html>
<html>
<head>
  <style>
    * {
      color: blue;
    }
  </style>
</head>
<body>
  <h1>Hello World!</h1>
  <p>This is a paragraph.</p>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
</body>
</html>
```

# 선택자와 선언부의 구조

- h1 요소에 32픽셀 글씨크기를 적용하기:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h1 {
      font-size: 32px;
    }
  </style>
</head>

<body>
  <h1>Hello World!</h1>
  <p>This is a paragraph.</p>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
</body>
</html>
```

# 선택자와 선언부의 구조

- 클래스 이름이 intro인 요소에 굵은 글씨체를 적용하기

```
<!DOCTYPE html>
<html>

<head>
  <style>
    .intro {
      font-weight: bold;
    }
  </style>
</head>

<body>
  <h1 class="intro">Welcome to my website</h1>
  <p>This is an introduction.</p>
  <ul>
    <li>Item 1</li>
    <li class="intro">Item 2</li>
  </ul>
</body>

</html>
```

# 선택자와 선언부의 구조

- ID 이름이 header인 요소에 회색 배경색을 적용하기:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    #header {
      background-color: gray;
    }
  </style>
</head>
<body>
  <div id="header">
    <h1>Welcome to my website</h1>
    <nav>
      <ul>
        <li>Home</li>
        <li>About</li>
        <li>Contact</li>
      </ul>
    </nav>
  </div>
  <p>This is the main content.</p>
</body>
</html>
```

# 선택자와 선언부의 구조

- h1, h2, h3 요소에 Arial 글꼴과 대체 글꼴을 적용하기:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h1, h2, h3 {
      font-family: Arial, sans-serif;
    }
  </style>
</head>
<body>
  <h1>Welcome to my website</h1>
  <h2>About us</h2>
  <h3>Contact information</h3>
  <p>This is some text.</p>
</body>
</html>
```

# 선택자 예제 코드

- 이 코드에서는 다양한 선택자를 사용하여 HTML 요소에 스타일을 적용하고 있습니다.
- "h1" 요소 선택자
  - HTML 파일 내의 모든 "h1" 태그에 적용됩니다.
- ".intro" 클래스 선택자
  - HTML 파일 내의 "class" 속성 값이 "intro" 인 모든 요소에 적용됩니다.
- "#main" ID 선택자
  - HTML 파일 내의 "id" 속성 값이 "main"인 요소에 적용됩니다.
- "div > p" 자식 선택자
  - "div" 태그의 자식으로 바로 아래에 있는 "p" 태그에 적용됩니다.
- "h2 + p" 인접 형제 선택자
  - "h2" 태그 바로 다음에 있는 "p" 태그에 적용됩니다.
- "h2 ~ p" 일반 형제 선택자
  - "h2" 태그 다음에 있는 모든 "p" 태그에 적용됩니다.

```
<!DOCTYPEhtml>
<html>
  <head>
    <style>
      /* 요소 선택자 */
      h1 {
        color: red;
      }

      /* 클래스 선택자 */
      .intro {
        font-size: 20px;
      }

      /* ID 선택자 */
      #main {
        background-color: gray;
      }

      /* 자식 선택자 */
      div > p {
        text-decoration: underline;
      }

      /* 인접 형제 선택자 */
      h2 + p {
        color: blue;
      }

      /* 일반 형제 선택자 */
      h2 ~ p {
        font-weight: bold;
      }
    </style>
  </head>
  <body>
    <div id="main">
      <h1>Welcome to my website!</h1>
      <p class="intro">Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
      <h2>Section 1</h2>
      <p>This is the first paragraph in section 1.</p>
      <p>This is the second paragraph in section 1.</p>
      <h2>Section 2</h2>
      <p>This is the first paragraph in section 2.</p>
      <p>This is the second paragraph in section 2.</p>
    </div>
  </body>
</html>
```

# 선택자와 선언부의 구조

- 모든 선택자를 사용한 예제

<https://gist.github.com/ej31/cb09db00d88696a6dd71e2a3018179ac> > 기본선택자 1

<https://gist.github.com/ej31/08da3dc673af74ef21a816d7e8e45a28> > 기본선택자 2

<https://gist.github.com/ej31/bbf07f81d3d433fe60ab540fb96395e5> > 가상선택자



# 유용한 속성들

- display
  - 사용 분야: 레이아웃 디자인
  - 요소의 표시 방법을 설정하는 속성입니다. block, inline, inline-block, flex 등 다양한 값으로 요소를 배치하고 정렬할 수 있습니다.
- position
  - 사용 분야: 레이아웃 디자인
  - 요소의 위치를 설정하는 속성입니다. static, relative, absolute, fixed 등 다양한 값으로 요소를 원하는 위치에 배치할 수 있습니다.
- background
  - 사용 분야: 배경 디자인
  - 요소의 배경 색상, 이미지, 위치 등을 설정하는 속성입니다. background-color, background-image, background-position, background-size, background-repeat 등 다양한 값을 활용하여 배경을 꾸밀 수 있습니다.

# 유용한 속성들

- border
  - 사용 분야: 테두리 디자인
  - 요소의 테두리를 설정하는 속성입니다. border-width, border-style, border-color 등 다양한 값을 조합하여 요소의 테두리를 꾸밀 수 있습니다.
- font
  - 사용 분야: 텍스트 디자인
  - 요소 내부의 글꼴, 크기, 굵기, 색상 등을 설정하는 속성입니다. font-family, font-size, font-weight, color 등 다양한 값을 활용하여 텍스트를 꾸밀 수 있습니다.
- transition
  - 사용 분야: 애니메이션 효과
  - 요소의 상태 변화를 부드럽게 처리하는 속성입니다. transition-property, transition-duration, transition-timing-function, transition-delay 등 다양한 값을 조합하여 요소의 상태 변화에 애니메이션 효과를 부여할 수 있습니다.

# 유용한 속성들

- box-shadow
  - 사용 분야: 그림자 효과
  - 요소에 그림자 효과를 추가하는 속성입니다. box-shadow를 사용하여 요소의 그림자 위치, 크기, 색상, 흐림 정도 등을 설정할 수 있습니다.
- flex
  - 사용 분야: 레이아웃 디자인
  - 요소의 유연한 박스 모델을 구현하는 속성입니다. display: flex를 사용하여 요소를 컨테이너로 지정하고, flex-direction, justify-content, align-items 등 다양한 값을 조합하여 요소를 배치하고 정렬할 수 있습니다.
- box-sizing
  - 사용 분야: 박스 모델 디자인
  - 요소의 크기 계산 방법을 설정하는 속성입니다. content-box는 요소의 크기가 콘텐츠 영역만을 기준으로 계산되며, border-box는 테두리와 패딩까지 포함한 전체 박스 영역을 기준으로 계산됩니다.

# 유용한 속성들

- transform
  - 사용 분야: 변환 효과
  - 요소의 변환 효과를 구현하는 속성입니다. transform: translate(), transform: rotate(), transform: scale() 등 다양한 값을 조합하여 요소의 위치, 회전, 크기 등을 조작할 수 있습니다.
- opacity
  - 사용 분야: 투명도 조절
  - 요소의 투명도를 설정하는 속성입니다. opacity: 0.5와 같이 값이 0~1 사이의 실수로 지정되며, 0에 가까울수록 요소가 투명해지고, 1에 가까울수록 불투명해집니다.

# 유용한 속성들

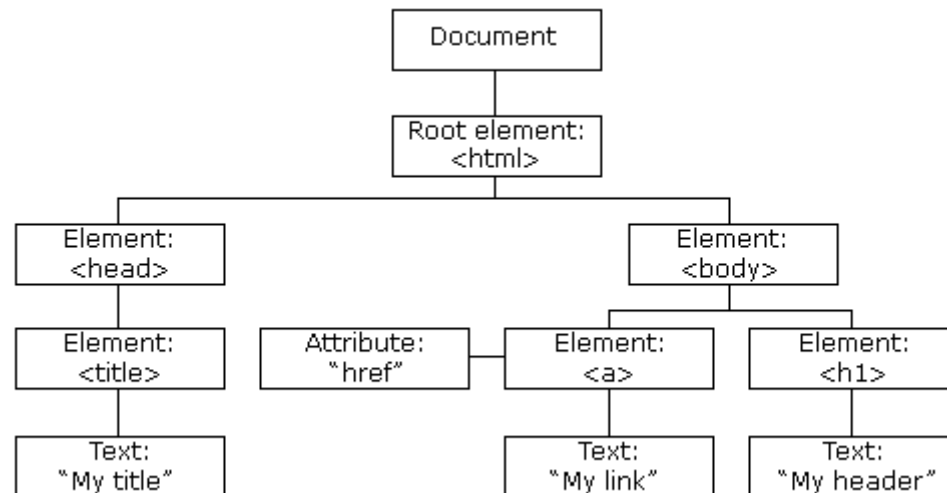
- overflow
  - 사용 분야: 내용 넘침 처리
  - 요소의 내용이 자신의 영역을 넘칠 때 처리 방법을 설정하는 속성입니다.  
overflow: hidden을 사용하여 요소의 영역을 벗어나는 내용을 숨기거나,  
overflow: scroll을 사용하여 스크롤 바를 추가할 수 있습니다.
- z-index
  - 사용 분야: 층위 조절
  - 요소의 층위를 조절하는 속성입니다. 값이 큰 요소가 값이 작은 요소 위에 위치하며, position 속성이 static이 아닌 요소에만 적용됩니다.

# 유용한 속성들

- <https://gist.github.com/ej31/2efa24a3251aa40e209719baac3c7d44>
  - 유용한 속성들 예제 1
- <https://gist.github.com/ej31/45e4066be814637e2f25abdc4ab465ad>
  - 유용한 속성들 예제 2

# DOM Tree Object (Document Object Model)

- 위 이미지에서, HTML 문서의 각 요소는 트리 구조에서 노드(node)로 표현됩니다. 요소 노드는 자식(child) 노드와 부모(parent) 노드를 가질 수 있으며, 각 노드는 DOM의 객체(object)로 나타납니다. 또한, 속성(attribute) 노드와 텍스트(text) 노드도 DOM tree에서 표현됩니다.
- 이와 같이 DOM tree는 HTML 문서의 구조를 계층적으로 표현하므로, 각 요소와 노드에 대한 구조를 이해하는 데 도움이 됩니다.



# DOM Tree Object 설명 상세

- DOM(Document Object Model)은 HTML, XML, SVG 등의 문서를 객체로 나타낸 모델입니다. DOM은 문서의 구조화된 표현을 제공하며, 프로그래밍 언어가 DOM 구조에 접근할 수 있는 방법을 제공합니다. 이를 통해 JavaScript와 같은 스크립트 언어를 사용하여 문서의 내용, 구조, 스타일 등을 변경할 수 있습니다.
- DOM tree는 문서의 각 요소(element), 속성(attribute), 텍스트(text) 등을 계층적으로 구조화한 트리 구조입니다. 각 요소는 객체(object)로 표현되며, 트리의 루트(root)는 document 객체입니다.



# DOM Tree Object 설명 상세

## 1.요소(element)

- 요소는 HTML 문서의 태그를 나타냅니다. 예를 들어, <div>, <p>, <a> 등의 태그가 요소에 해당합니다. 요소는 태그의 이름을 가지고 있으며, 속성을 가질 수도 있습니다. 각 요소는 자식 요소와 부모 요소를 가질 수 있으며, 트리 구조에서 노드(node)로 표현됩니다.

## 2.속성(attribute)

- 속성은 요소의 특성을 나타냅니다. 예를 들어, href, class, id 등의 속성이 있습니다. 속성은 요소 내에 키-값 쌍으로 표현되며, 트리 구조에서 요소 노드의 하위 노드로 표현됩니다.

# DOM Tree Object 설명 상세

## 3. 텍스트(text)

- 텍스트는 HTML 문서의 요소 내에 포함된 일반적인 텍스트를 나타냅니다. 예를 들어, `<p>Hello, world!</p>`와 같은 코드에서 "Hello, world!"가 텍스트에 해당합니다. 텍스트는 요소의 내용으로 표현되며, 트리 구조에서 요소 노드의 자식 노드로 표현됩니다.

## 4. 객체(object)

- 각 요소, 속성, 텍스트는 객체로 표현됩니다. 객체는 해당 요소, 속성, 텍스트의 속성과 메서드를 가지고 있으며, JavaScript와 같은 스크립트 언어를 사용하여 객체의 속성과 메서드에 접근할 수 있습니다.
- DOM tree는 HTML 문서의 각 요소를 계층적으로 구조화하여 표현하므로, JavaScript와 같은 스크립트 언어를 사용하여 요소, 속성, 텍스트를 동적으로 변경할 수 있습니다. 이를 통해 웹 페이지의 동적인 UI를 구현할 수 있으며, 사용자와 상호작용하는 다양한 기능을 구현할 수 있습니다.

# DOM Tree Object 설명 상세

- 위의 예제 코드에서, h1, div, p, ul, li와 같은 HTML 요소는 DOM tree에서 각각 노드 (node)로 표현됩니다. 이러한 노드는 다른 노드와 계층적 관계를 가지며, 이를 통해 문서의 구조를 나타냅니다.

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Tree 예제</title>
</head>
<body>
  <h1>제목</h1>
  <div>
    <p>첫 번째 단락입니다.</p>
    <p>두 번째 단락입니다.</p>
    <ul>
      <li>리스트 1</li>
      <li>리스트 2</li>
      <li>리스트 3</li>
    </ul>
  </div>
  <div>
    <p>세 번째 단락입니다.</p>
    <p>네 번째 단락입니다.</p>
  </div>
</body>
</html>
```

# DOM Tree Object 설명 상세


- 이제 이 예제에서 "두 번째 단락"을 선택하여 스타일을 변경하는 예제를 작성해보겠습니다. 다음은 JavaScript를 사용하여 "두 번째 단락" 요소를 선택하는 코드입니다.

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Tree 예제</title>
</head>
<body>
  <h1>제목</h1>
  <div>
    <p>첫 번째 단락입니다.</p>
    <p>두 번째 단락입니다.</p>
    <ul>
      <li>리스트 1</li>
      <li>리스트 2</li>
      <li>리스트 3</li>
    </ul>
  </div>
  <div>
    <p>세 번째 단락입니다.</p>
    <p>네 번째 단락입니다.</p>
  </div>
  <script>
    var secondParagraph = document.getElementsByTagName("p")[1];
    // secondParagraph.style.color = "red"; // 이 부분을 웹 브라우저 콘솔에 입력하면 색이 변한다.
  </script>
</body>
</html>
```

# DOM Tree Object 설명 상세

- 위 코드에서, `getElementsByTagName("p")`는 문서 내의 모든 p 요소를 선택한 후, [1]은 두 번째 p 요소를 선택합니다. 이렇게 선택된 요소는 변수에 할당됩니다.

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Tree 예제</title>
</head>
<body>
  <h1>제목</h1>
  <div>
    <p>첫 번째 단락입니다.</p>
    <p>두 번째 단락입니다.</p>
    <ul>
      <li>리스트 1</li>
      <li>리스트 2</li>
      <li>리스트 3</li>
    </ul>
  </div>
  <div>
    <p>세 번째 단락입니다.</p>
    <p>네 번째 단락입니다.</p>
  </div>
  <script>
    var secondParagraph = document.getElementsByTagName("p")[1];
    // secondParagraph.style.color = "red"; // 이 부분을 웹 브라우저 콘솔에 입력하면 색이 변한다.
  </script>
</body>
</html>
```



# 스타일 시트 종류

- 내부 스타일 시트 (Internal Style Sheet): HTML 문서 안에 <style> 태그를 사용하여 작성한 스타일 시트로, 해당 HTML 문서에서만 사용됩니다.
- 외부 스타일 시트 (External Style Sheet): 확장자가 .css인 파일로, HTML 문서에서 <link> 태그를 사용하여 로드하여 사용하는 스타일 시트입니다. 여러 개의 HTML 문서에서 공유하여 사용할 수 있습니다.
- 인라인 스타일 (Inline Style): HTML 요소의 style 속성에 직접 작성한 스타일 시트로, 해당 요소에서만 사용됩니다.

# 스타일 시트 예제

- 내부 스타일 시트 예제
  - <https://gist.github.com/ej31/b2fd61f03c20f7ad8a1a13841cd174e8>
- 외부 스타일 시트 예제
  - <https://gist.github.com/ej31/f8de8dadbbac0bcebf991a5b5507c799>
- 인라인 스타일 예제
  - <https://gist.github.com/ej31/7727c65e514e4efbfa5a20db543d1230>

# 박스 모델

- CSS 박스모델은 HTML 요소들이 차지하는 공간을 사각형 박스로 정의하는 개념입니다. 각 박스는 여백(margin), 테두리(border), 내부여백(padding), 콘텐츠(content)로 구성됩니다.

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS Box Model Example</title>
  <style>
    .box {
      width: 200px;
      height: 100px;
      margin: 20px;
      padding: 10px;
      border: 2px solid #333;
      background-color: #f7f7f7;
      box-sizing: border-box;
    }
  </style>
</head>
<body>
  <div class="box">Box Model Example</div>
</body>
</html>
```

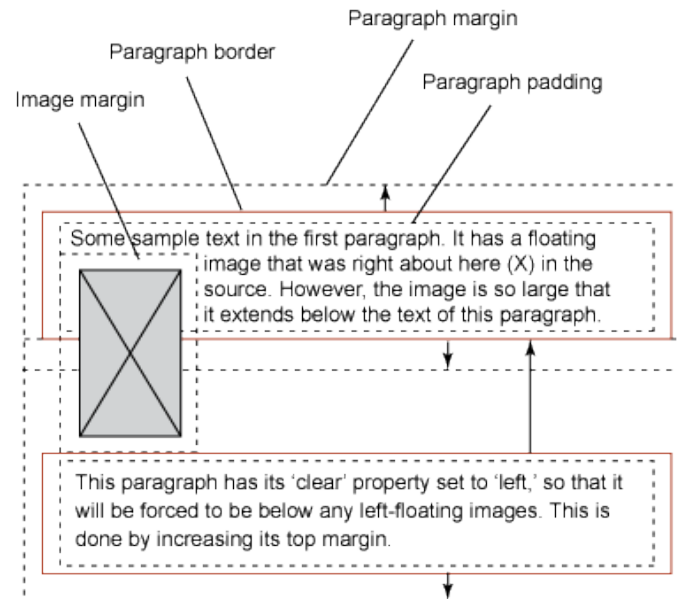


# 박스 모델

- 박스 모델을 지정하는 요소의 값은 생략이 가능하며 그 상세는 예제 코드와 같다.
  - <https://gist.github.com/ej31/dea6d8e0d709bb9632fe6097659309c1>

# 플로팅과 클리어링

- 플로팅은 요소를 떠있는 상태로 배치하는 것입니다. 요소가 플로팅 상태일 때, 다른 요소들이 그 주위를 감싸는 형태로 배치됩니다. 주로 이미지나 텍스트와 같은 콘텐츠를 배치할 때 사용됩니다.
- `floating:left` 혹은 `floating:right` 를 사용할 경우 `display`는 무시됩니다.
- 과거에는 레이아웃을 구성할 때 플로팅을 이용하기도 했습니다.
- 현재는 Flexbox, Grid를 사용합니다.

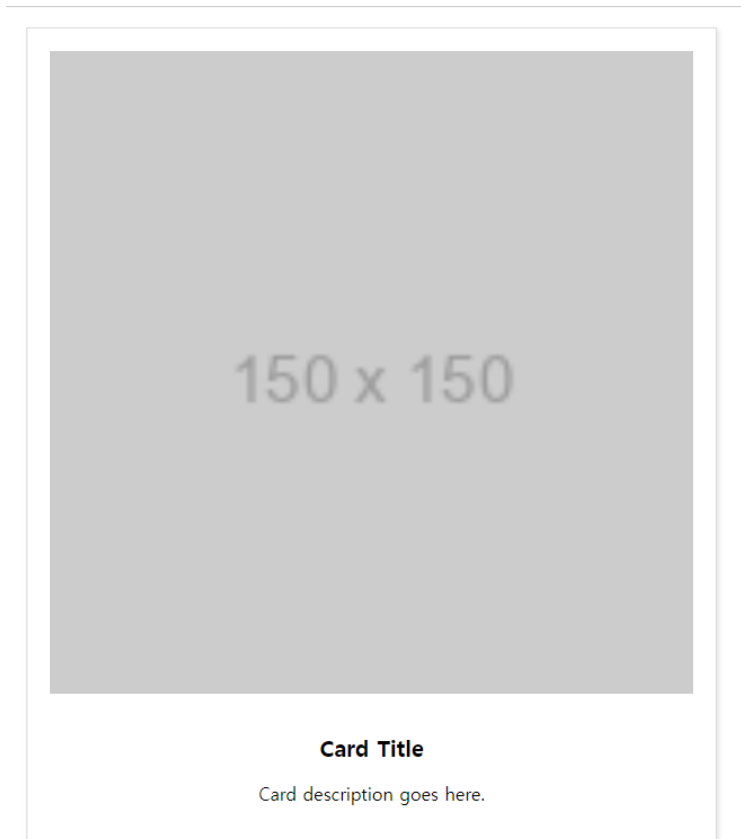


# 플로팅과 클리어링

- 클리어링은 플로팅된 요소의 영향을 해제하는 기술입니다. 클리어링을 사용하면, 다음 요소가 플로팅된 요소의 영역에 들어가지 않습니다.
- 예제코드
  - <https://gist.github.com/ej31/fae3e38f44fc7f9d082974b72577d70a>

# 플로팅과 클리어링 예제문제

- 아래 카드를 만들어보세요



# 포지셔닝

- CSS의 position 속성을 이용하여 요소의 위치를 조절할 수 있습니다. position 속성은 요소의 위치를 결정하는 방법으로, static, relative, absolute, fixed, sticky 등의 값을 가질 수 있습니다.
- static: 요소의 위치를 문서 흐름에 따라 정렬합니다.
- relative: 요소를 문서 흐름에 따라 정렬하되, top, right, bottom, left 속성을 이용하여 요소의 위치를 이동시킬 수 있습니다.
- absolute: 부모 요소를 기준으로 top, right, bottom, left 속성을 이용하여 요소를 이동시킵니다.
- fixed: 브라우저 창을 기준으로 top, right, bottom, left 속성을 이용하여 요소를 이동시킵니다. z-index와 요소가 겹치는 경우 축이 엉킬 수 있다.
- sticky: 스크롤 영역 내에서 요소를 고정시킵니다.

# 포지셔닝

- 예제코드

- <https://gist.github.com/ej31/fae3e38f44fc7f9d082974b72577d70a>

```
<div style="position: relative; border: 1px solid black; padding: 10px; height: 800px;">
  <p style="background-color: yellow;">첫 번째 문장</p>
  <p style="background-color: blue; position: absolute; top: 50px; left: 50px;">두 번째 문장</p>
  <p style="background-color: green; position: absolute; top: 150px; left: 150px;">세 번째 문장</p>
  <p style="background-color: red; position: fixed; top: 20px; right: 20px;">네 번째 문장</p>
  <p style="background-color: purple; position: sticky; top: 10px;">다섯 번째 문장</p>
  <p style="background-color: orange; position: relative; z-index: 1;">여섯 번째 문장</p>
</div>
```

# 플렉스 박스

- CSS Flexbox는 웹 페이지의 레이아웃을 구성하기 위한 유연하고 강력한 방법 중 하나입니다. Flexbox는 부모 요소와 자식 요소 간의 관계를 기반으로 하는 레이아웃 모델입니다.
- Flexbox는 다음과 같은 속성을 사용하여 부모 요소를 조정합니다.
  - display: flex
    - Flexbox를 사용하려면 부모 요소에 display: flex를 적용해야 합니다.
  - flex-direction
    - Flexbox는 행 방향 (가로) 또는 열 방향 (세로)으로 배치할 수 있습니다. flex-direction 속성을 사용하여 방향을 지정할 수 있습니다.
  - justify-content
    - Flexbox는 부모 요소 내에서 자식 요소를 수평으로 정렬할 수 있습니다. justify-content 속성을 사용하여 수평 정렬을 지정할 수 있습니다.
  - align-items
    - Flexbox는 부모 요소 내에서 자식 요소를 수직으로 정렬할 수 있습니다. align-items 속성을 사용하여 수직 정렬을 지정할 수 있습니다.
  - flex-wrap
    - Flexbox는 자식 요소가 부모 요소 내에서 너무 많은 경우에 자식 요소를 여러 행 또는 열로 나눌 수 있습니다. flex-wrap 속성을 사용하여 요소의 줄 바꿈을 지정할 수 있습니다.

# 플렉스 박스

- Flexbox의 자식 요소는 다음과 같은 속성을 사용하여 조정할 수 있습니다.
  - flex-grow
    - 자식 요소를 늘리는 정도를 지정합니다.
  - flex-shrink
    - 자식 요소를 축소하는 정도를 지정합니다.
  - flex-basis
    - 자식 요소의 기본 크기를 지정합니다.
  - order
    - 자식 요소의 순서를 지정합니다.
- Flexbox는 반응형 웹 디자인과 함께 사용하기에 이상적입니다. Flexbox를 사용하면 디바이스의 크기와 방향에 따라 레이아웃이 동적으로 조정됩니다.
- 예제코드
  - <https://gist.github.com/ej31/ef11472e28670d382987a5a031e03cd6>



# 플렉스 박스

- Flexbox의 자식 요소는 다음과 같은 속성을 사용하여 조정할 수 있습니다.
  - flex-grow
    - 자식 요소를 늘리는 정도를 지정합니다.
  - flex-shrink
    - 자식 요소를 축소하는 정도를 지정합니다.
  - flex-basis
    - 자식 요소의 기본 크기를 지정합니다.
  - order
    - 자식 요소의 순서를 지정합니다.
- Flexbox는 반응형 웹 디자인과 함께 사용하기에 이상적입니다. Flexbox를 사용하면 디바이스의 크기와 방향에 따라 레이아웃이 동적으로 조정됩니다.
- 예제코드
  - <https://gist.github.com/ej31/ef11472e28670d382987a5a031e03cd6>