1 CSS-1

4.CSS 이해하기

1) CSS 소개

CSS와 HTML
CSS는 간단히 이야기하면 HTML(마크업 언어)을 꾸며주는 표현용 언어입니다.
HTML이 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. 그러나 분명히 독립된 언어!
CASCADING STYLE SHEETS, STYLE SHEETS는 말 그대로 스타일 규칙을 정의

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.
아래 참고 링크를 방문하면 하나의 HTML이 다양한 CSS를 통해 바뀌는 것을 확인할 수 있습니다.
http://www.csszengarden.com/

2) CSS 문법과 적용

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

- 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; }"

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 */
```

CSS의 적용
CSS와 문서를 연결하는 방법은 4가지가 있습니다.

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.

 내용

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 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.

3) 선택자 1

요소 선택자

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

선택자 중에 가장 기본이 되는 선택자이며, 태그 선택자라고도 합니다.
요소 선택자는 위 코드처럼 선택자 부분에 태그 이름이 들어갑니다.
문서 내에 선택자에 해당하는 모든 요소에 스타일 규칙이 적용됩니다.

그룹화

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

선택자는 쉼표를 이용해서 그룹화를 할 수 있습니다.
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;}
```

4) 선택자 2

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>
```

위 코드처럼 <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>
  <dd><span class="html css">JS</span>는 문서의 동작을 나타냅니다.</dd>
</dl>
```

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

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

1. .기호가 아닌 #기호 사용
2. 태그의 class 속성이 아닌 id 속성을 참조
3. 문서 내에 유일한 요소에 사용
4. 구체성

가장 큰 차이점은 class와 달리 id는 문서 내에서 유일해야 한다는 점입니다.
클래스 선택자는 여러 요소에 같은 클래스를 넣고 같은 규칙을 적용 할 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.

5) 선택자 3

선택자의 조합

/* 요소와 class의 조합 */
p.bar { ... } 이 경우에는 <p>이면서 class 속성에 bar가 있어야 적용됩니다.
/* 다중 class */
.foo.bar { ... } 이 경우에는 class 속성에 foo와 bar가 모두 있어야 적용됩니다.
/* id와 class의 조합 */
#foo.bar { ... } 이 경우에는 id가 foo이며 class가 bar인 요소에 적용됩니다.

속성 선택자

단순 속성으로 선택

속성 선택자는 대괄호를 이용해서 선언하며 대괄호 안에 속성 이름이 들어갑니다.
요소에 해당 이름의 속성이 있다면 해당 사항이 적용됩니다.

p[class] { color: silver; } <p>이면서 class 속성이 있는 요소이면 color: silver 규칙이 적용
p[class][id] { text-decoration: underline; }
<p>이면서 class 속성과 id 속성이 함께 있어야 text-decoration: underline 규칙이 적용

```
<p class="foo">Hello</p>
<p class="bar">CSS</p>
<p class="baz" id="title">HTML</p>
```

위 CSS 코드는 요소 선택자와의 조합으로 이루어진 코드입니다.
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.

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

ex)

```
<p class="color hot">red</p>
<p class="cool color">blue</p>
<p class="colorful nature">rainbow</p>
```

```
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 */
```

6) 문서 구조 관련 선택자

문서 구조를 이용한 선택자는 3가지 있습니다.
자손 선택자와 자식 선택자 그리고 인접 선택자입니다.
문서 구조를 잘 이해하셨다면 쉽게 예측할 수 있습니다.

자손 선택자

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

자손 선택자는 선택자 사이에 아무 기호없이 그냥 공백으로 구분을 합니다.
이 선택자는 의 자손 요소인 를 선택하는 선택자 입니다.

자식 선택자

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

자식 선택자는 선택자 사이에 닫는 꺽쇠 기호(>)를 넣습니다.
꺽쇠 기호와 선택자 기호 사이에는 공백은 있거나 없어도 상관이 없습니다.
이 선택자는 의 자식 요소인 <h1>를 선택하는 선택자 입니다.

인접 형제 선택자

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

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

```
body > div table + ul { ... }
/* body 요소의 자식인 div 요소의 자손인 table 요소 바로 뒤에 인접한 ul 요소 선택! */
```

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

예시

```
<head>
  <meta charset="utf-8">
  <title>quiz</title>
  <style media="screen">
     .wrap .item.i2 {color: red;}
     .wrap .list .i2 {color: red;}
     div > .list .i2 {color: red;}
     strong + .list .i2 {color: red;} (X!!!)
  </style>
</head>

<body>
  <div class="wrap">
```

```
    <strong> title </strong>
    <p> description </p>

    <ul class="list">

      <li class="item i1"> item 1 </li>
      <li class="item i2"> item 2 </li>
      <li class="item i3"> item 3 </li>

    </ul>
  </div>
</body>
```

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.
예를 들어, <p>가 있다고 가정하겠습니다.
이 <p>에 마우스 커서를 올렸을 때만 특정 스타일을 주고 싶다고 한다면 어떻게 해야 할까요?
가상 클래스가 없다면 이런 과정을 거치게 됩니다.

1. 임의의 클래스 선택자를 선언하여 특정 스타일 규칙을 만든다.
2. p 요소에 커서가 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 여러 패턴에 대해서 미리 정의해놓고, 가상 클래스로 제어할 수 있게 했습니다.
위처럼 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.
이 수업에서는 가장 기초적이고 대표적인 가상 클래스만을 설명해 드리기 때문에 아래 링크를 참고 직접 공부하시길 바랍니다.
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 선택자 입니다.

- :first-child : 첫 번째 자식 요소 선택
- :last-child : 마지막 자식 요소 선택

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

<ul>
  <li>HTML</li>
```

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

아래와 같이 동작!

```
<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 되는 요소들은 다 사용이 가능합니다.

- :focus: 현재 입력 초점을 가진 요소에 적용 - tab처럼 선택 받을 때
- :hover: 마우스 포인터가 있는 요소에 적용 - 마우스 올렸을 때
- :active: 사용자 입력으로 활성화된 요소에 적용 - 클릭할 때, 눌렀을 때 순간적으로 활성화

```
a:focus { background-color: yellow; }
a:hover { font-weight: bold; }
a:active { color: red; }
```

8) 가상 선택자 2

가상 요소(pseudo element)
가상 요소는 HTML 코드에 존재하지 않는 구조 요소에 스타일을 부여할 수가 있습니다.
가상 요소도 가상 클래스처럼 문서 내에 보이지 않지만, 미리 정의한 위치에 삽입되도록 약속이 되어있습니다.
선언 방법은 가상 클래스와 같게 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.
강의에서는 가상 클래스와 마찬가지로 기본적인 가상 요소만 설명해 드리고 마치도록 하겠습니다.
다른 가상 요소들에 대한 자세한 내용은 아래 링크를 참고해주세요.
https://developer.mozilla.org/ko/docs/Web/CSS/Pseudo-elements

```
p::before { content: "###" } 가장 앞에 요소를 삽입
```

p::after { content: "!!!" } 가장 뒤에 요소를 삽입 (자식 요소들 중에서 가장 마지막에 위치하도록, /* after를 사용할 때는 반드시 position이나 display 같은 속성이 먼저 들어가야 함. */)
p::first-line { color: yellow; } 요소의 첫 번째 줄에 있는 텍스트 (모니터 가로 해상도에 따라 요소가 포함하는 내용이 변동됨)
p::first-letter { font-size: 3em; } 블록 레벨 요소의 첫 번째 문자

before와 after 가상 요소는 애초에 내용이 없는 상태로 생성되기 때문에 내용을 넣기 위해 content 속성을 이용해야 합니다.

9) 구체성

구체성
요소를 선택하는 데는 여러 방법이 있습니다.
따라서 서로 다른 선택자를 이용해 같은 요소를 선택할 수도 있습니다.
만약 같은 요소를 선택하는 서로 다른 규칙들이 상반된 스타일을 가지고 있다면 어떻게 표현이 될까요?
두 규칙은 모두 <h1>을 선택하게 됩니다.
하지만 두 규칙이 지정하는 스타일은 서로 다릅니다.
그렇다면 <h1>은 어떻게 표현이 될까요?
<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.
위와 같이 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 */
```

인라인 스타일
지금까지 선택자의 구체성에 대해 살펴봤습니다.
인라인 스타일의 구체성 값은 1, 0, 0, 0이며 규칙들 중 가장 큰 구체성을 갖기 때문입니다.
p#page { color: red;} 0 1 0 1
Lorem impusm dolor sit. 1 0 0 0 blue

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

10) 상속

상속되는 속성

```
h1 { color: gray; }
<h1>Hello, <em>CSS</em></h1>
```

위 코드에서 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 상식적으로 상속되지 않는다는 것을 알고 계시면 됩니다.

상속되는 속성의 구체성

```
* { color: red; }
h1#page { color: gray; }
<h1 id="page">Hello, <em>CSS</em></h1>
```

위 코드에서는 전체 선택자를 이용해 color: red를 적용하고 id 선택자를 이용해 color: gray를 선언했습니다.
전체 선택자의 구체성은 0, 0, 0, 0 이며 id 선택자의 구체성은 0,1,0,1 입니다.
그렇다면 에는 어떤 color가 적용될까요?
color: red가 적용되는데 그 이유는 바로 상속된 속성은 아무런 구체성을 가지지 못하기 때문입니다.
상속된 CSS는 구체성을 가지지 못하기 때문에((0, 0, 0, 0)도 아님) 구체성을 가진 전체 선택자의 스타일이 적용이된다.

11) 캐스케이딩

cascading 규칙

cascading에는 다음과 같이 3가지 규칙이 있습니다.

1. 중요도(!important)와 출처
2. 구체성
3. 선언 순서

위에서의 출처는 CSS 출처를 의미합니다.
CSS 출처는 제작자(개발자)와 (일반)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.

스타일이 적용되는 방식은 생각보다 간단합니다.
모든 스타일은 아래의 규칙에 따라 단계적으로 적용됩니다.

1. 스타일 규칙들을 모아서 중요도!important가 명시적으로 선언되었는지에 따라 분류합니다.
    1. 중요도가 명시적으로 선언된 규칙들은 그렇지 않은 규칙들보다 우선합니다.

        2. 중요도가 있는 규칙들끼리는 아래 다른 규칙들을 적용받습니다.
2. 스타일 규칙들을 출처에 따라 분류합니다.
        1. 제작자(개발자) 스타일 규칙이 사용자 에이전트 스타일 규칙보다 우선합니다.
3. 스타일 규칙들을 구체성에 따라 분류합니다.
        1. 구체성이 높은 규칙들이 우선합니다.
4. 스타일 규칙들을 선언 순서에 따라 분류합니다.
        1. 뒤에 선언된 규칙일수록 우선합니다.

12) 선택자 정리

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

5.단위, 배경, 박스모델

1) 속성-정의와구문 -- 레퍼런스 보는 법

https://www.w3schools.com/ 간단하게 스펙과 예제를 확인하는데 용이합니다.
https://developer.mozilla.org/ko/개발적 버그나 추가적인 스펙에 대해서 확인하는데 용이합니다.
https://www.w3.org/ CSS 속성을 정의하는 W3C에서 제공하는 사이트이므로 가장 정확합니다.

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 웹사이트의 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 인쇄단위입니다.
웹 화면에 인쇄용 문서를 위한 스타일을 적용할 때 유용하게 사용할 수 있습니다.
그러나 사용하는 기기의 해상도에 따라 차이가 있어 W3C에서도 pt는 웹개발 시 권장하는 단위가 아닙니다.
예를 들면 Windows에서는 9pt = 12px, Mac에서는 9pt = 9px 로 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. (1em == 16px)

- rem
  root의 font-size를 기준으로 값을 환산합니다.

- vw
  viewport의 width값을 기준으로 1%의 값으로 계산됩니다.

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 정의되어있는 키워드를 이용해 색상을 표현할 수 있습니다.
  ** 참고 : transparent는 투명을 나타내는 키워드 **

- **16 진법** ex. **#RRGGBB**
  6자리의 16진수(0-9, A-F)는 각각 두 자리씩 세 가지 색상을 나타냅니다.
  첫 번째 두 자리는 적색(RR), 가운데 두 자리는 녹색(GG), 마지막 두 자리는 청색(BB)을 의미합니다.
  각 자리의 알파벳은 대소문자를 구분하지 않습니다.

- **16 진법** 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. 마지막값을 입력하지 않으면 애초에 이 식이 먹지 않아요!

```
<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>
```

4) 속성-background

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

- background-color 기본 값 : transparent

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

- background-image 기본 값 : none / url(img/promotion.png)

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 기본 값 : 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.

< 속성 값 >
repeat x, y축 으로 모두 반복합니다.
repeat-x x 축 방향으로만 반복합니다.
repeat-y y 축 방향으로만 반복합니다.
no-repeat 이미지를 반복하지 않습니다.

- background-position 기본 값 : 0% 0% 요소에서 배경 이미지의 위치를 지정하는 속성입니다. x축, y축으로부터의 위치를 지정할 수 있으며, 값의 선언 순서는 x축, y축으로부터의 간격입니다. 만일 한쪽만 지정된다면 나머지는 중앙 값(center)으로 적용됩니다.

< 속성 값 >
% 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.

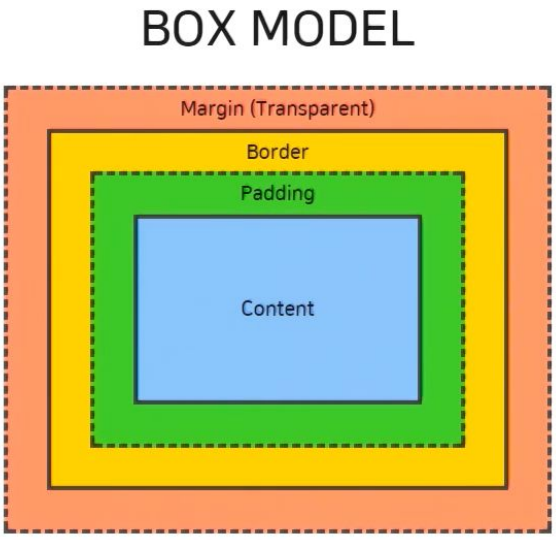
- background-attachment 기본 값 : scroll

화면 스크롤에 따른 배경 이미지의 움직임 여부를 지정하는 속성입니다.

< 속성 값 >
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.

```
<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>
```

5) 속성-boxmodel



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.

6) 속성-border

> border: border-width border-style border-color|initial|inherit;

```
/* 속성 */
border-width: 1px; /* 두께  <length> | thin | medium | thick */
border-style: solid; /* 종류 none | hidden | dotted | dashed | solid | double | groove | ri
border-color: #000; /* 색상 */

/* 축약형 */
border: 1px solid #000; /* 단축 속성 */
border-left: 6px solid red; /* 보더 왼쪽 */
border-width: 6px; /* 보더 두께 */
border-width: 6px 3px; /* 보더 상하, 좌우 두께 */
border-width: 6px 3px 4px; /* 보더 상, 우(좌), 하 두께 */
border-width: 6px 3px 4px 1px; /* 보더 상, 우, 하, 좌 두께 */
```

border 관련 속성

- **border-width** 기본 값 : 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];

〈속성 값〉
키워드 thin, medium, thick
단위 px, em, rem ... (% , 정수 단위 사용불가)

- **border-style** 기본 값 : none 선의 모양을 지정하는 속성입니다. border-top-style, border-bottom-style, border-right-style, border-left-style을 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.

〈속성 값〉
none border를 표시 하지 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** 기본 값 : 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.

```
border-width: 10px
border-style: solid;
border-color: #000;

/* 축약형 */
border: 10px solid #000;
```

7) 속성-padding

```
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 */
```

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

8) 속성-margin

margin 속성 : 기본 값 : 0

〈속성 값〉
length 고정값으로 지정합니다. (ex. px, em ....)
percent 요소의 width에 상대적인 크기를 지정합니다.
auto 브라우저에 의해 계산된 값이 적용 됩니다.
padding과 마찬가지로 축약하여 사용할 수 있고, 상하, 좌우에 대해서 값이 같으면 하나로 합하여 사용할 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. (기준이 되는 width 가 꼭 있어야한다) 기준이 있어야 정렬을 하잖아요?

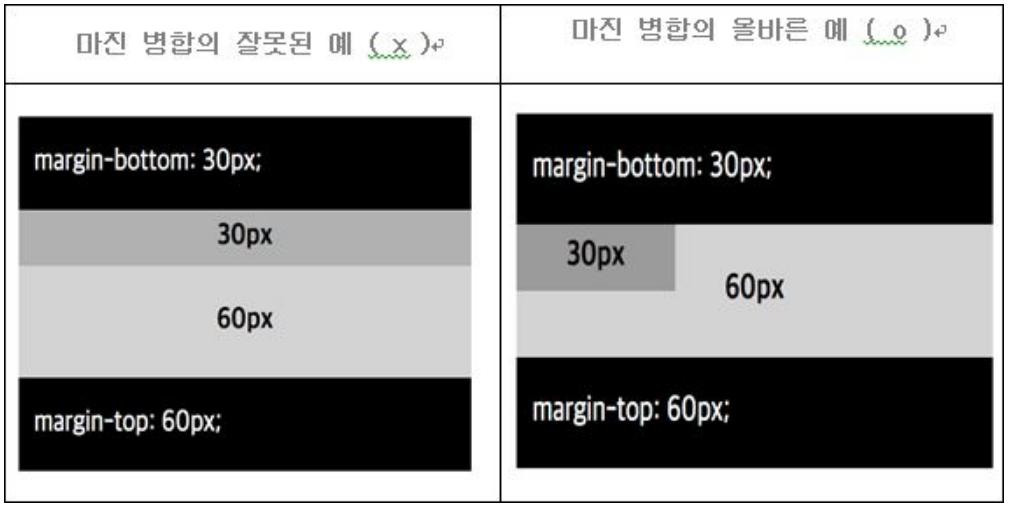
아래 코드를 살펴봅시다.
margin-left: auto; 왼쪽을 기준으로 auto값이 적용되어 오른쪽 끝에 가는 것을 확인할 수 있음
margin-right: auto; 오른쪽을 기준으로 auto값이 적용되어 왼쪽 끝에 가는 것을 확인할 수 있음
margin: 0 auto; 가로축 정렬 시
margin auto: 오잉 근데 이것도
margin auto auto; 이것도 비슷한데 심화개념이라 아직 ㄴㄴ

좌우의 margin이 모두 auto로 적용 되었다면, 브라우저는 요소가 가질수 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 좀더 심화적인 개념이 필요하므로 기초에서는 수평 정렬에 대한 개념을 확실히 잡아두시길 바랍니다.

margin collapse(마진 병합) → 다시 듣기
마진 병합은 인접한 두 개 이상의 수직 방향 박스의 마진이 하나로 합쳐지는 것을 의미합니다.
마진 병합이 다음 세가지의 경우에 일어납니다.

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
    1. 부모 요소의 상단 마진과 첫 번째 자식 요소의 상단 마진 병합이 일어납니다.
    2. 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 병합을 활용하여 첫 번째와 두 번째 컴포넌트의 조합이 다양한 경우 여백을 다르게 사용할 수 있습니다.
<h1>Margin Collapse</h1>
* What do you think? *
h1 { margin: 0 0 20px 0; }
div { margin-top: 40px; }
p { margin-top: 30px ;}

40px임 왜?

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 적용할 수 있을까? 라는 생각을 가지고 코딩하는 자세는 매우 중요합니다. margin과 padding은 px과 같은 고정적인 단위 외에도 %라는 상대적인 단위를 사용할 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 관계없이 모두 요소의 width 값(가로축)을 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.

사람이야기 잘 외우는 방법 ㅎㅎ
패딩 지방, 보더 겉피부, 마진 사람과 사람사이의 거리

10) 속성-width

요소의 가로 크기를 지정하는 width 속성은 인라인 레벨 요소를 제외한 모든 요소에 적용됩니다.
블록&인라인 강의에서 다룬 블록 레벨 요소와 인라인 레벨 요소의 특징을 되짚어보시기 바랍니다.

width 속성
기본 값 : auto

〈속성 값〉

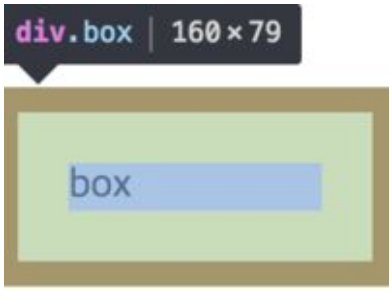
auto	브라우저에 의해 자동으로 계산하여 적용합니다. * 요소의 레벨 기본 특성에 따라 다르게 동작합니다.
length	고정값으로 지정합니다. (ex. px, em ....)

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 어떻게 160px가 된 걸까요?
앞선 설명에서 언급한 바와 같이 width는 padding, border 영역에 대해서 영향을 받기 때문입니다.

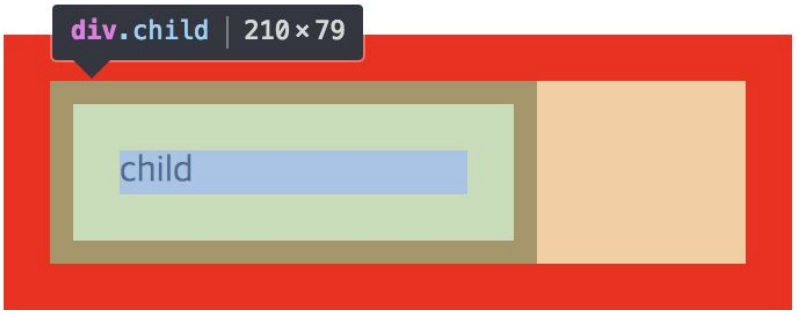


식으로 나타내면,
100px *content* + (20px * 2) *padding* + (10px * 2) *border* = 160px(박스 사이징이) 가 된 것입니다.

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;
}
```

위와 같이 코드를 적용 했을 때 .child의 크기는 어떻게 적용될까요?



정답은 210px 입니다.
우선, padding (20px * 2) + border (10px * 2) = 60px 입니다.
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.

box값은 padding+border+content값

11) 속성-height

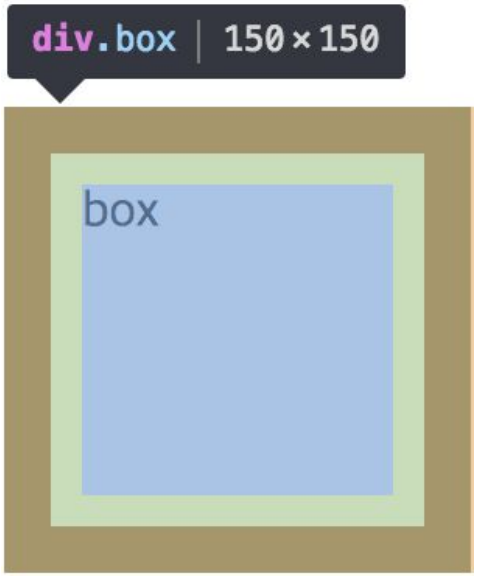
height 속성
기본 값 : auto

< 속성 값 >
auto 브라우저 자동으로 계산하여 적용합니다. * 기본적으로 컨텐츠 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.

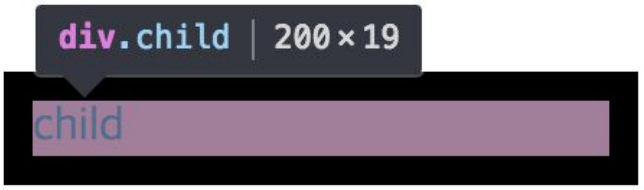


식으로 나타내면, 100px *content* + (10px * 2) *padding* + (15px * 2) *border* = 150px 가 된 것입니다.

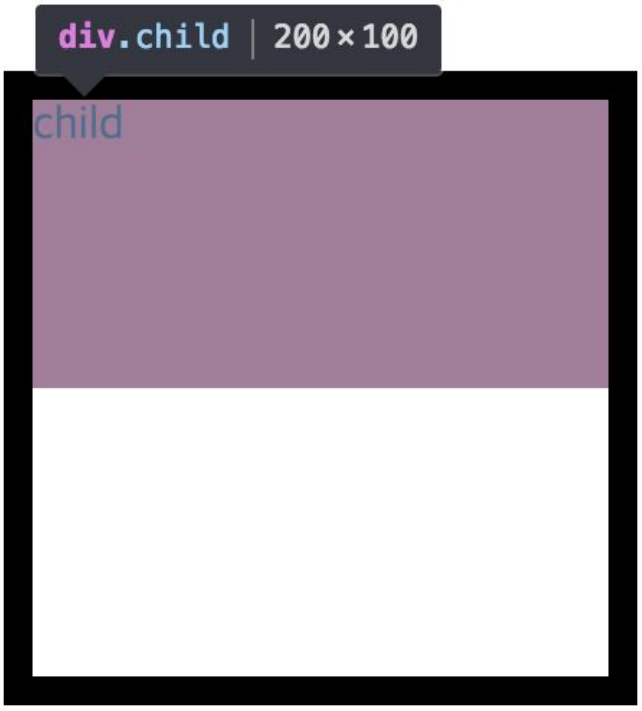
%를 이용해서도 크기를 지정 할 수 있습니다.

```
<div class="parent">
 <div class="child">
   child
 </div>
</div>
.parent {
 width: 200px;
 border: 10px solid black;
}
.child {
 height: 50%;
 background: red;
}
```

위와 같이 코드를 적용 했을때, .child 의 높이는 어떻게 적용 될까요?



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.

12) 속성-boxmodel정리

지금까지 *< 단위, 배경, 박스모델 >* 챕터를 통해서 박스 모델은 content, padding, border, margin의 총 4가지 영역으로 나누어진다는 걸 배웠습니다.
추가적으로 content 영역의 너비는 width, 높이는 height을 통해서 제어 할 수 있으며,
width와 height 그리고 padding과 border을 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 기억하시기 바랍니다.