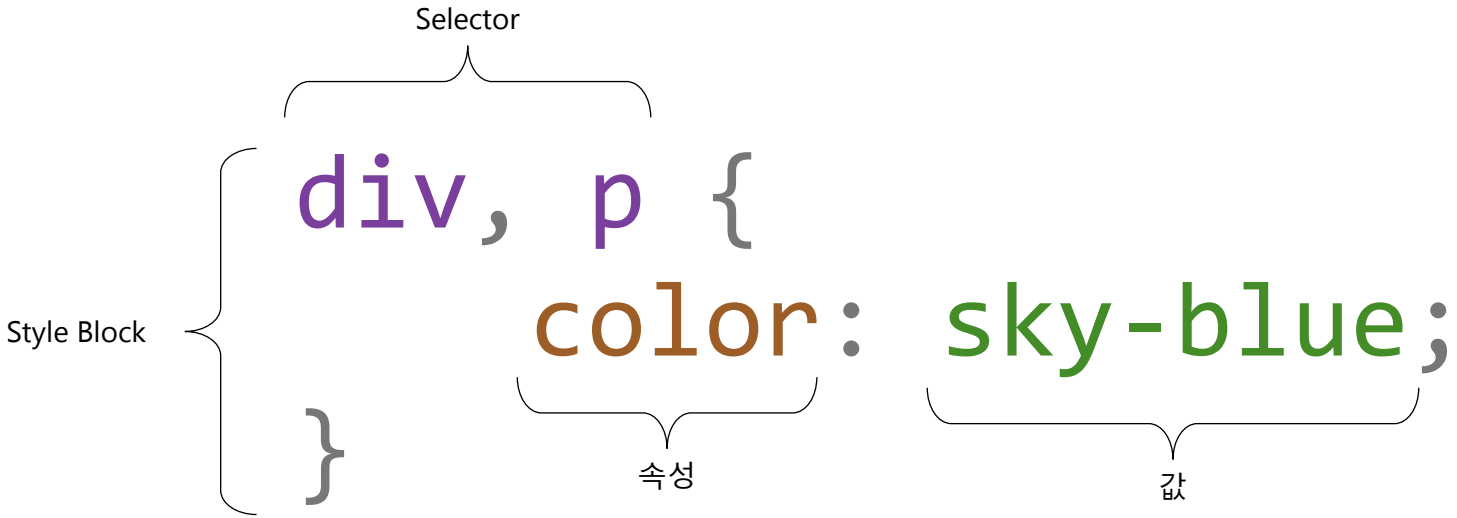
CSS

김상훈

CSS

- Cascading Style Sheet
- 문서의 표현을 기술하는 스타일 시트 언어



CSS 적용 방법

Inline

```
<div style="display: none;">
</div>
```

- 각 태그마다 스타일을 모두 적어야 하므로 관리가 쉽지 않음
- 우선 순위가 가장 높기 때문에 특별한 경우가 아니면 쓰지 말아야 함
- 메일 본문 스타일을 만들 때 주로 사용

Embedded

```
<head>
  <style>
    div {display: none;}
  </style>
</head>
<body>
```

- 보통 head안에 Style을 감싸서 넣음
- CSS가 간단한 페이지 일 경우 사용
- 사용자에게 초기 로딩시 보여주는 화면을 구성할 때 사용

External

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

- 별도의 CSS 파일로 분리
- 관심사 분리와 재사용이 가능
- 가장 많이 사용하는 방법

CSS 상속

- CSS 상속

```
<div style="color: yellow;"">
  <div></div>
</div>
```

부모 Element에 적용된 Style이
자식 Element에 적용됨

Element의 속성 > 부모 Element의 속성 > 브라우저 기본 속성

- 일부 속성에 대해서만 상속

상속 되는 속성
visibility
font
color
line-height
text-align
…

상속 되지 않는 속성
margin
padding
border
width
height
display
…

CSS Selector

CSS 셀렉터

- CSS를 적용할 요소를 지칭

```
<p>안녕하세요.</p>
<p id="hello">안녕하세요.</p>
<p class="hello">안녕하세요.</p>
<input type="button" value="안녕하세요">
```

안녕하세요.

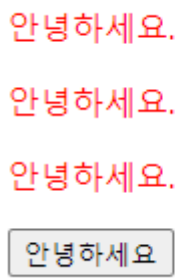
안녕하세요.

안녕하세요.

안녕하세요

CSS 태그 셀렉터

```
<p>안녕하세요.</p>
<p id="hello">안녕하세요.</p>
<p class="hello">안녕하세요.</p>
<input type="button" value="안녕하세요">


p {
    color: red;
}
```

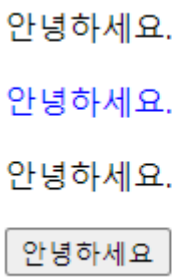


CSS ID 셀렉터

- Id 값 앞에 #을 붙여서 선택
- 우선순위 때문에 쓰지 않는 것이 좋음

```
<p>안녕하세요.</p>
<p id="hello">안녕하세요.</p>
<p class="hello">안녕하세요.</p>
<input type="button" value="안녕하세요">
```

```
#hello {
    color: blue;
}
```



CSS 클래스 셀렉터

- Class앞에 .을 붙여서 선택
- 가장 많이 쓰이는 셀렉터

```
<p>안녕하세요.</p>
<p id="hello">안녕하세요.</p>
<p class="hello">안녕하세요.</p>
<input type="button" value="안녕하세요">
```

```
.hello {
    color: green;
}
```

안녕하세요.

안녕하세요.

안녕하세요.

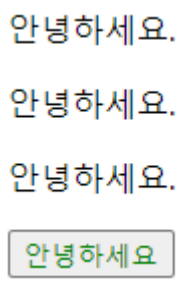
안녕하세요

CSS 속성 셀렉터

- 대괄호 안에 속성명 = "값" 형태로 사용
- 대괄호 안에 속성명 형태로도 사용 가능

```
<p>안녕하세요.</p>
<p id="hello">안녕하세요.</p>
<p class="hello">안녕하세요.</p>
<input type="button" value="안녕하세요">
```

```
[type="button"] {
    color: green;
}
```



CSS 셀렉터 – 부모자식 관계

- html 문서는 부모 자식 관계가 있음
- header와 h1, h2는 부모-자식 관계
- h1과 h2는 형제 관계

```
<header>
    <h1>조 루소</h1>
    <h2>안소니 루소</h2>
</header>
```

CSS 후손 셀렉터

- 부모와 후손을 선택 시 공백 사용
- 여러 개의 셀렉터를 적용할 때 콤마(,)로 구분

```
<header>
    <h1>조 루소</h1>
    <h2>안소니 루소</h2>
</header>
```

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

조 루소

안소니 루소

```
header h1,
header h2 {
    color: red;
}
```

조 루소

안소니 루소

CSS 자식 셀렉터

- 부모와 자식 선택 시 꺽쇠(>) 사용

```
<header>
    <p>Avengers</p>
    <div><p>End game</p></div>
</header>
```

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

Avengers

End game

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

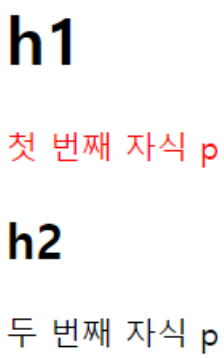
Avengers

End game

CSS 바로 뒤 형제 셀렉터

- 바로 뒤 형제 선택 시 사용

```
<h1>h1</h1>
<p>첫 번째 자식 p</p>
<h2>h2</h2>
<p>두 번째 자식 p</p>
```

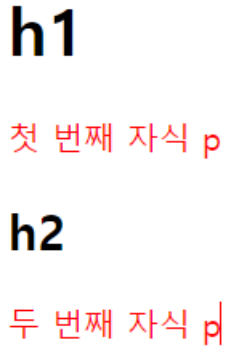



```
h1 + p {
    color:red;
}
```

CSS 뒤에 오는 모든 형제 셀렉터

- 뒤에 오는 모든 형제 선택 시 ~ 사용

```
<h1>h1</h1>
<p>첫 번째 자식 p</p>
<h2>h2</h2>
<p>두 번째 자식 p</p>
```




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

CSS 전체 셀렉터

- 모든 엘리먼트 선택 시 * 사용
- 성능에 좋지 않아 남발하지 않는 것이 좋음

```
<header>
    <p>Avengers</p>
    <div><p>End game</p></div>
</header>
```

Avengers

End game

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

CSS 유사 클래스 셀렉터

- Element가 특별한 상태일 때를 선택
- 마우스가 올라가 있거나, 선택되어 있거나 등

```
<header>
    <p>Avengers</p>
    <div><p>End game</p></div>
</header>

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

Avengers ← 마우스가 올라가면 색이 변경됨

End game

유사 클래스 셀렉터	설명
:hover	마우스 오버
:active	선택된 상태
:focus	포커스가 있을 때
:checked	체크 상태일 때
:disabled	사용 불가능일 때
:first-child, :last-child	해당 요소 중 첫 번째, 마지막
:nth-child(n)	해당 요소 중 n 번째
:nth-of-type(n)	해당 요소 중 n 번째 엘리먼트
:not(셀렉터)	해당 요소가 아닌 것들

CSS 유사 Element 셀렉터

- Element 내용의 앞과 뒤에 내용을 삽입
- 가상요소이므로 블록 지정이 안됨

```
<div>캡틴  아메리카</div>
```

```
div::before {
    content: "마블's";
    color: red;
    margin-right: 10px;
}
div::after {
    content: "시리즈";
    color: blue;
    margin-left: 10px;
}
```

마블's 캡틴 아메리카 시리즈

- 존재하는 Element가 없는 가상의 요소를 선택
- 유사 클래스는 상태를 선택하고, 유사 Element는 선택적인 부분을 선택
- 주로 문자열을 지정할 수 있는 content 속성과 함께 사용
- 콜론 두 개(::) 와 함께 사용

유사 엘리먼트 셀렉터	설명
::first-letter	첫 번째 글자
::first-line	첫 번째 줄
::before	Element 내용의 앞
::after	Element 내용의 뒤
::selection	선택된 글자

CSS 우선 순위

- 하나의 Element에 여러 가지 셀렉터로 선택이 가능
- 각 셀렉터로 지정한 스타일 중 우선순위에 따라 적용
- 클래스 셀렉터 > 태그 셀렉터
- 셀렉터 우선순위 동등: 나중에 선언된 스타일 적용

```
<main>
    <h1 class="title">오드리 헵번</h1>
</main>
```

```
main h1 {
    color: green;
}

.title {
    color: red;
}
```

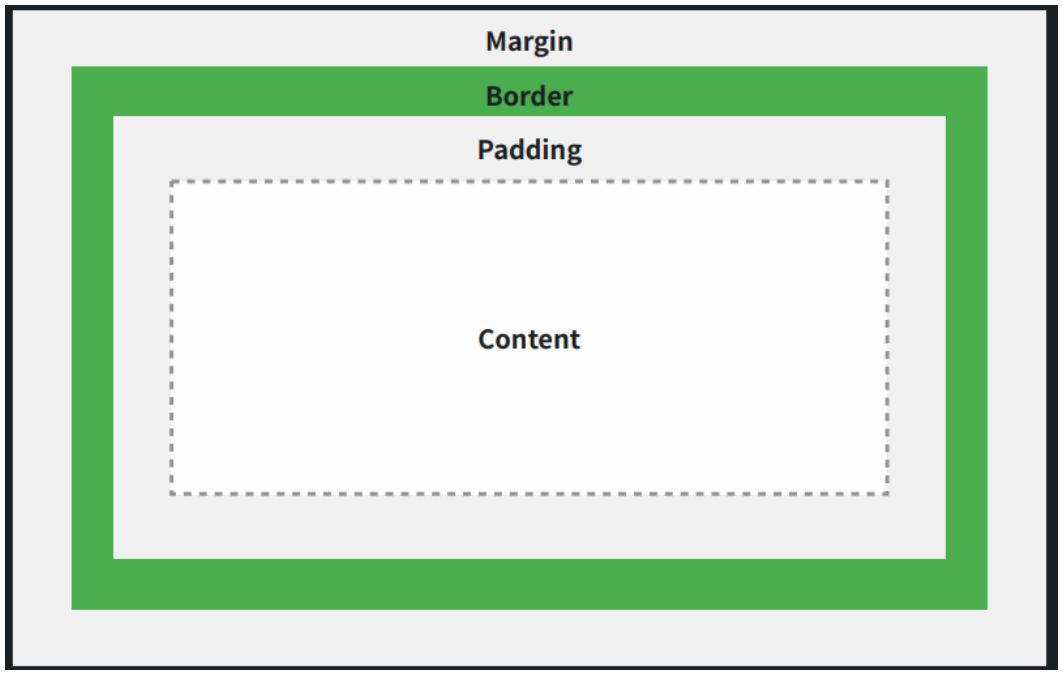
```
.title {
    color: red;
}

.title {
    color: green;
}
```

CSS 박스 모델

CSS 박스 모델

- 웹 문서의 내용을 박스 형태로 정의하는 방법이며, CSS 레이아웃의 기본이 되는 개념
- 브라우저가 Element를 렌더링 할 때 참고하는 값

박스 모델 구성 요소

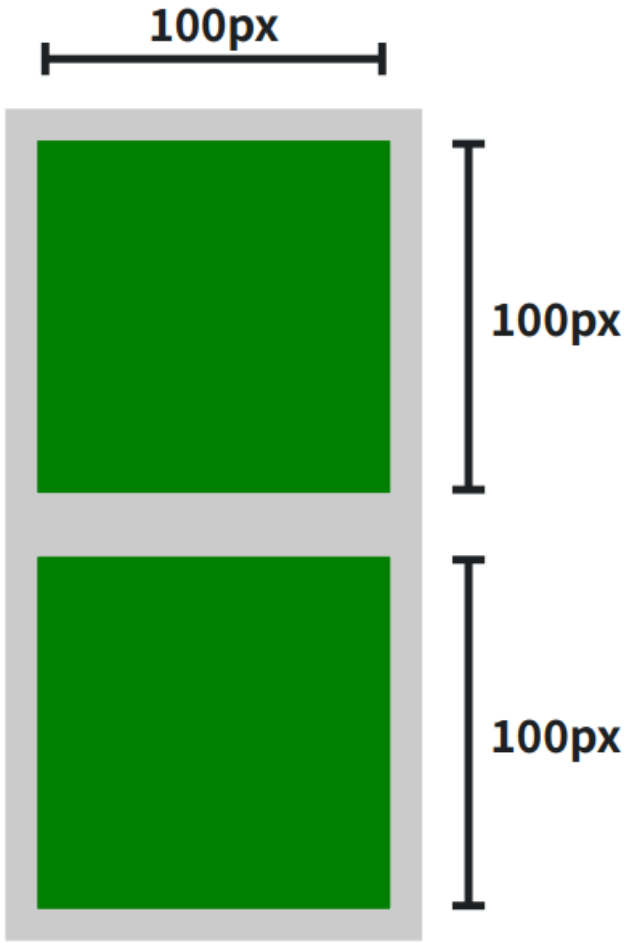


- Margin: 바깥 여백
- Border: 테두리
- Padding: 안쪽 여백
- Content: 내용

CSS 박스 모델: box-sizing

- content-box 내용을 기준으로 Element 기준을 잡음

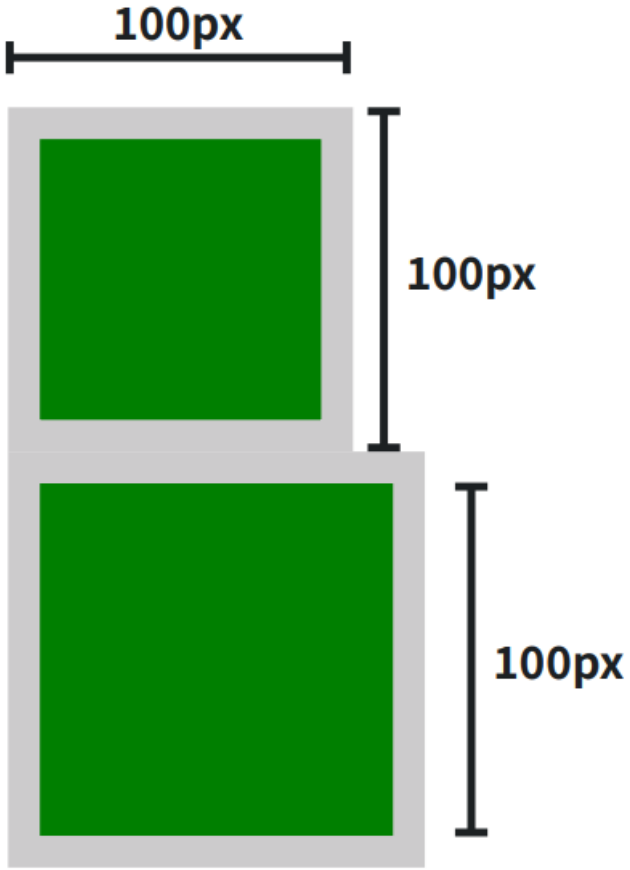
```
<div class="box border-box"></div>
<div class="box content-box"></div>


.box {
    border: 10px solid #ccc;
    width: 100px;
    height: 100px;
    background: green;
    box-sizing: content-box;
}
```



CSS 박스 모델: box-sizing

- border-box: 테두리부터 기준으로 Element 크기를 잡음

```
<div class="box border-box"></div>
<div class="box content-box"></div>


.box {
    border: 10px solid #ccc;
    width: 100px;
    height: 100px;
    background: green;
    box-sizing: content-box;
}

.border-box {
    box-sizing: border-box;
}
```



CSS 박스 모델: background-clip

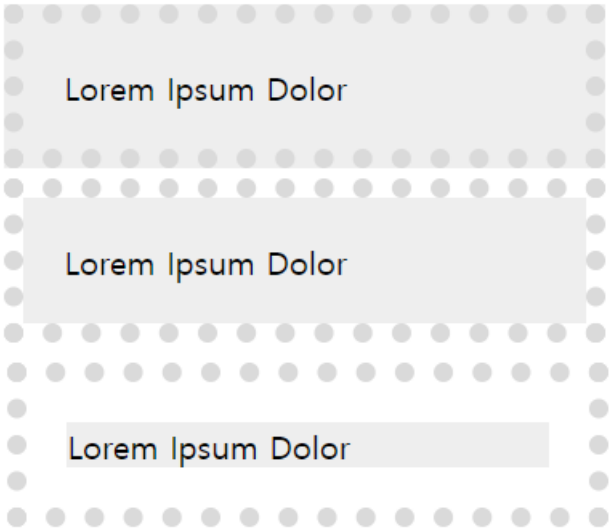
- 배경을 채워줄 범위 지정
- 박스 모델 구성요소를 살펴보기 좋은 속성

```
selector {
    background-clip: <background-clip 속성>;
}
```

Background-clip 속성	설명
border-box	테두리 영역과 그 안쪽 영역
padding-box	안쪽 여백 영역과 그 안쪽 영역
content-box	내용 영역과 그 안쪽 영역
initial	기본 값으로 설정
inherit	부모 요소의 속성값을 상속 받음

CSS 박스 모델: background-clip

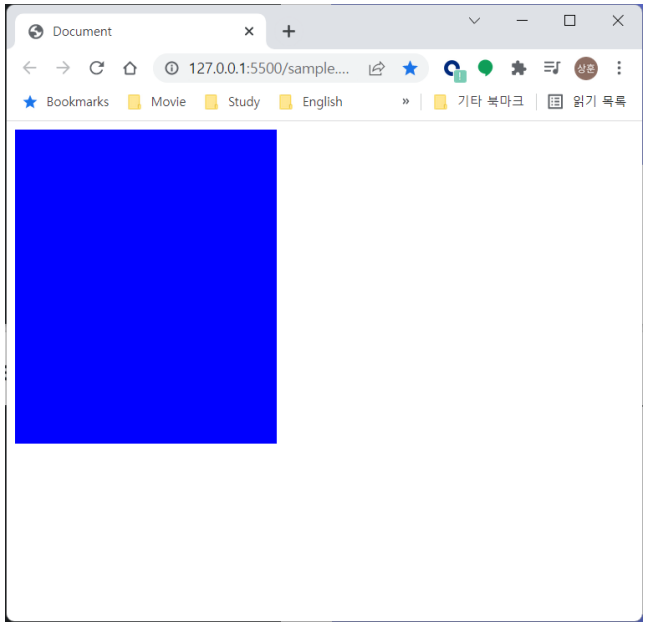
```
div {
    box-sizing: border-box;
    width: 300px;
    margin: 20px 0px;
    padding: 20px;
    border: 10px dotted #dadada;
    background-color: #eeeeee;
}
.borderbox {
    background-clip: border-box;
}
.paddingbox {
    background-clip: padding-box;
}
.contentbox {
    background-clip: content-box;
}
```



CSS 박스 모델: width, height

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

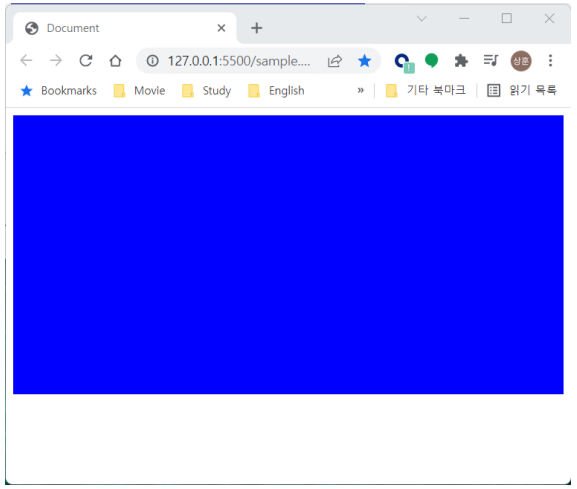
```css
.box {
    width: 250px;
    height: 300px;
    background: blue;
}
```



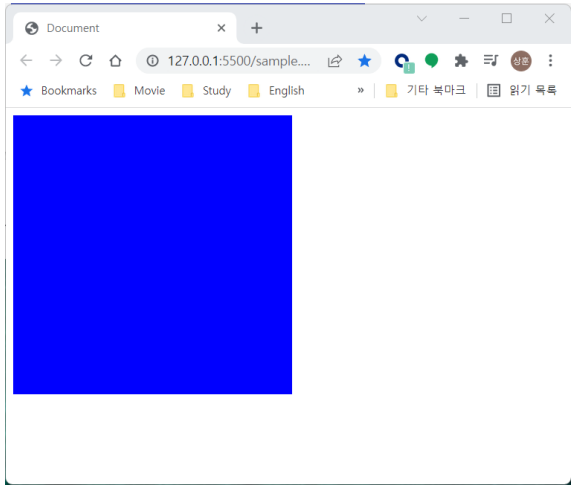
CSS 박스 모델: width, height

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

```css
.box {
    width: 100%;
    height: 300px;
    background: blue;
}
```

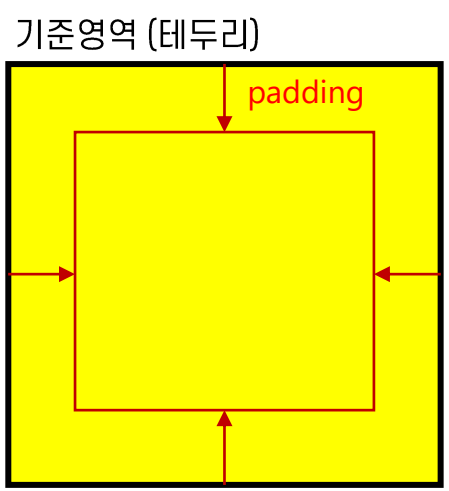


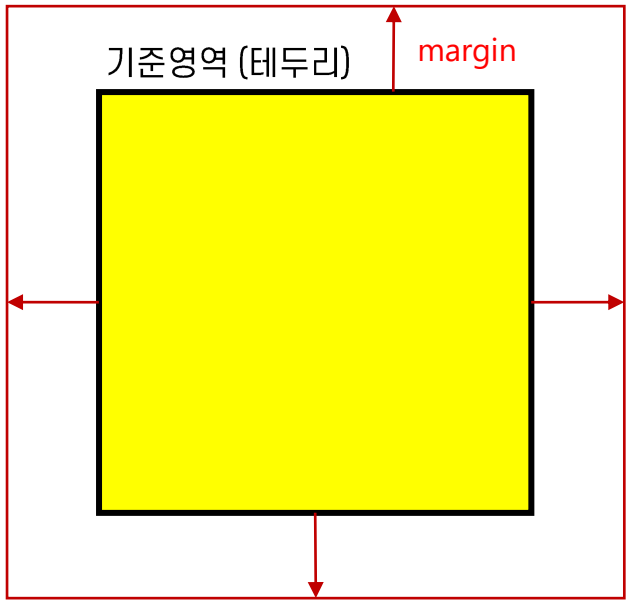

```css
.box {
    width: 100%;
    max-width: 300px;
    height: 300px;
    background: blue;
}
```



CSS 박스 모델: margin과 padding

- 정의된 테두리 요소 주위에 여백을 만듦



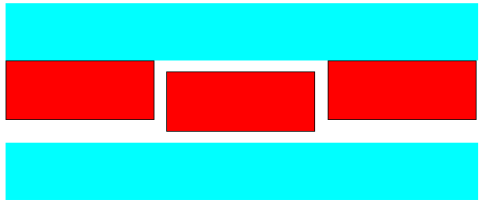


CSS 박스 모델: margin

```
<div style="width:100%; height:100%;">
    <div class="header"></div>
    <div class="box" style="float:left"></div>
    <div class="box center" style="float:left"></div>
    <div class="box" style="float:left"></div>
    <div class="footer" style="width:100%;float:left;"></div>
</div>
```

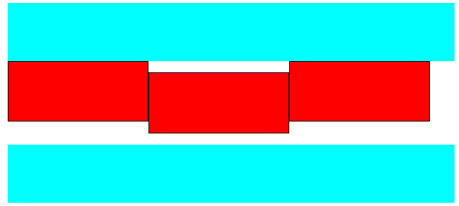
한 개 값: 상하좌우 값 설정

```
.center {
    margin: 10px;
}
```



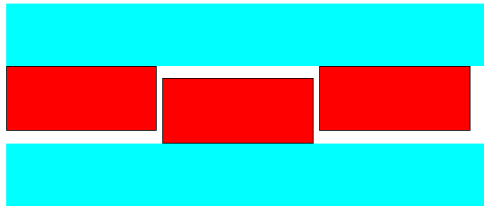
두 개 값: 상하, 좌우 값 설정

```
.center {
    margin: 10px 0;
}
```



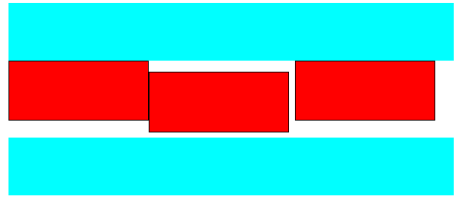
세 개 값: 상, 좌우, 하 값 설정

```
.target {
    margin: 10px 5px 0;
}
```



네 개 값: 상, 우, 하, 좌 값 설정

```
.target {
    margin: 10px 5px 5px 0;
}
```



CSS 박스 모델: margin

- 상하좌우 값을 따로 설정

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

```
.target {
    background: #00A2FF;
    margin-top: 10px;
}
```

```
.target {
    background: #00A2FF;
    margin-bottom: 10px;
}
```

```
.target {
    background: #00A2FF;
    margin-left: 10px;
}
```

```
.target {
    background: #00A2FF;
    margin-right: 10px;
}
```

?

CSS 박스 모델: padding

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

```css
.box {
    width: 400px;
    height: 200px;
    background: #5F6DC7;
    border: 1px solid #000;
}

.target {
    width: 100%;
    height: 100%;
    border: 1px solid #000;
    background: #00A2FF;
}
```

한 개 값: 상하좌우 값 설정

```css
.box {
    padding:10px;
}
```

두 개 값: 상하, 좌우 값 설정

```css
.box {
    padding:10px 0;
}
```

세 개 값: 상, 좌우, 하 값 설정

```css
.box {
    padding: 10px 50px 20px;
}
```

네 개 값: 상, 우, 하, 좌 값 설정

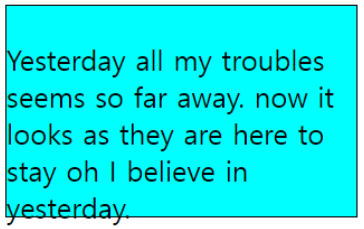
```css
.box {
    padding: 10px 50px 20px 0;
}
```

CSS 박스 모델: padding

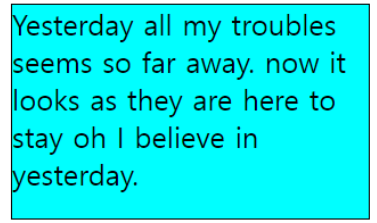
- 상하좌우 값을 따로 설정

```
<div class="box">Yesterday all my troubles seems so far away. now it looks as they are here to stay oh I believe in yesterday.</div>
```

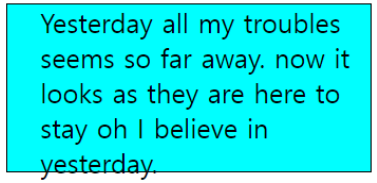
```
.box {
    padding-top: 20px;
}
```



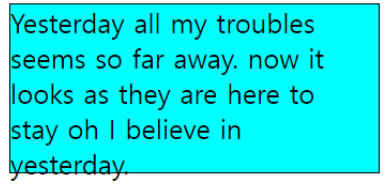

```
. box {
    padding-bottom: 10px;
}
```




```
. box {
    padding-left: 10px;
}
```




```
. box {
    padding-right: 10px;
}
```

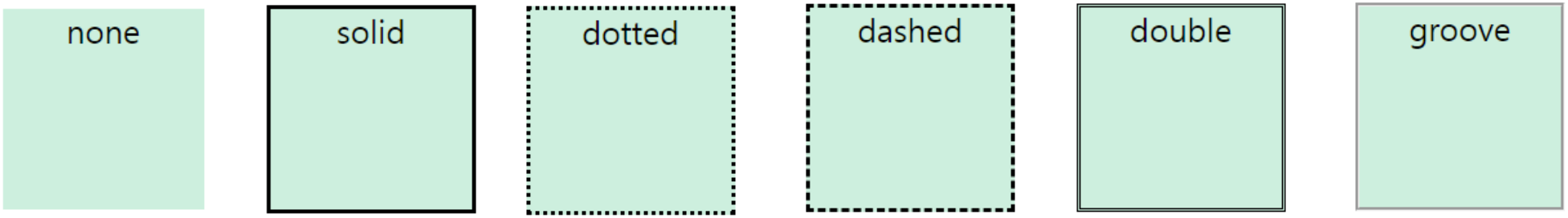


CSS 박스 모델: border

- 박스 모델의 테두리 지정

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


.box {
    width: 100px;
    height: 100px;
    background:#CDEFDE;
    border: solid;
}
```

Border-style 사용 가능 속성	설명
none	테두리가 없음. 기본 값
hidden	테두리를 감춤
solid	테두리를 실선으로 표시
dotted	테두리를 점선으로 표시
dashed	테두리를 짧은 직선으로 표시
groove	테두리를 창에 조각한 것 처럼 표시
inset	테두리가 창에 박혀 있는 것 처럼 표시
outset	테두리가 창에서 튀어나온 듯 표시
ridge	테두리가 창에서 튀어나온 듯 표시

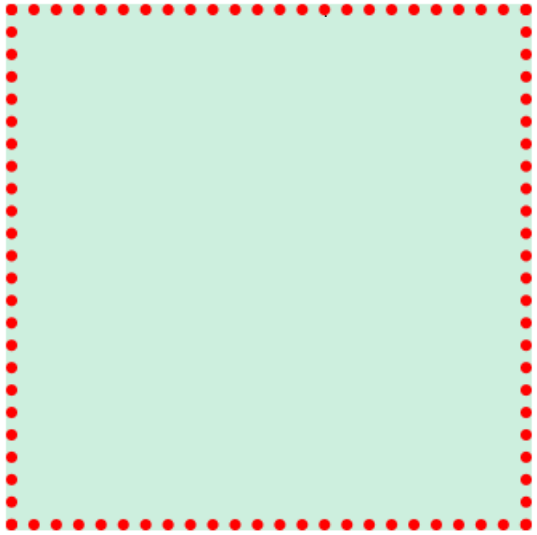


CSS 박스 모델: border

- 두께, 스타일, 색상 표시

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

```
.box {
    width: 200px;
    height: 200px;
    background:#CDEFDE;
    border: dotted red 5px;
}
```

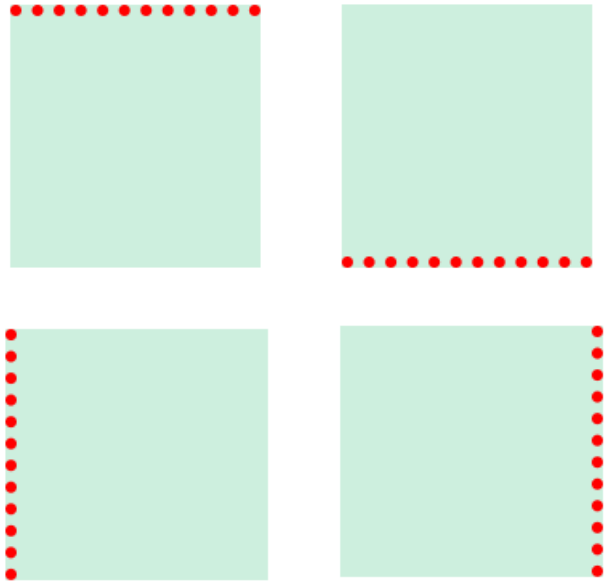


```
.box {
    width: 200px;
    height: 200px;
    background:#CDEFDE;
    border: dotted;
    border-color: red;
    border-width: 5px;
}
```

CSS 박스 모델: border

- 상, 하, 좌, 우 구분하여 표시

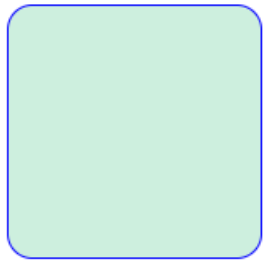
```
.box {
    border-top: dotted;
}

.box {
    border-bottom: dotted;
}

.box {
    border-left: dotted;
}

.box {
    border-right: dotted;
}
```



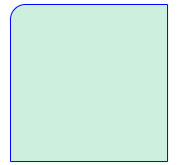
CSS 박스 모델: border

- Border-radius: 테두리를 둥글게 만듦

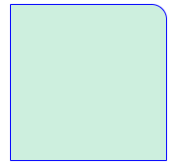
```
.box {
    width: 100px;
    height: 100px;
    background:#CDEFDE;
    border: solid blue 1px;
    border-radius: 10px;
}
```



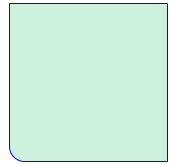
```
.box {
    border-top-left-radius: 10px;
}
```



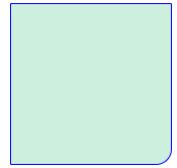
```
.box {
    border-top-right-radius: 10px;
}
```



```
.box {
    border-bottom-left-radius: 10px;
}
```



```
.box {
    border-bottom-right-radius: 10px;
}
```



CSS Display

Block 레벨 특성

- 한 라인을 모두 차지

```
<p>안녕하세요</p><p>안녕하세요</p><p>안녕하세요</p>
```

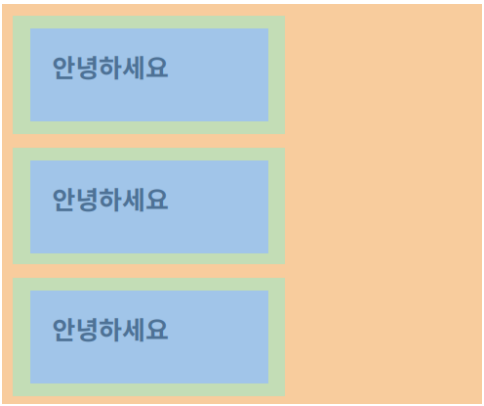
안녕하세요

안녕하세요

안녕하세요

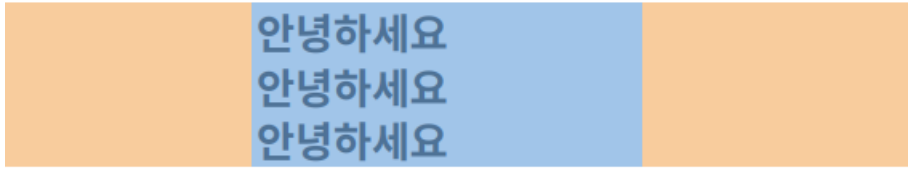
- width, height, margin, padding 속성 동작

```
p {
    margin: 10px;
    padding: 10px;
    width: 150px;
    height: 30px;
}
```



- margin: x auto;를 통해 가로 가운데 정렬

```
p {
    margin: 0 auto;
    width: 150px;
}
```



Inline 레벨 특성

- 여러 요소가 한 행에 있을 수 있음

```
<span>baseball</span><span>baseball</span>
```

baseballbaseball

- width, height, margin의 top, bottom 동작 안 함

```
span {
    margin: 10px;
    padding: 10px;
    width: 150px;
    height: 150px;
}
```

baseball baseball

- 부모 Element의 text-align:center로 가운데 정렬

```
<div class="parent">
    <span>rookie</span><span>rookie</span>
</div>

.parent {
    text-align: center;
}
```

baseballbaseball

CSS Display: Inline-block

- Block 레벨 요소도 여러 요소와 같이 한 행에 있을 수 있으며

```
<p>안녕하세요</p><p>안녕하세요</p><p>안녕하세요</p>
<span>baseball</span><span>baseball</span>
```

```
p, span {
    display: inline-block;
}
```

안녕하세요안녕하세요안녕하세요 baseballbaseball

- Inline 레벨 요소도 width, height, margin, padding을 모두 가질 수 있음

```
p, span {
    display: inline-block;
    margin: 10px;
    padding: 10px;
    width: 150px;
    height: 30px
}
```

안녕하세요 안녕하세요

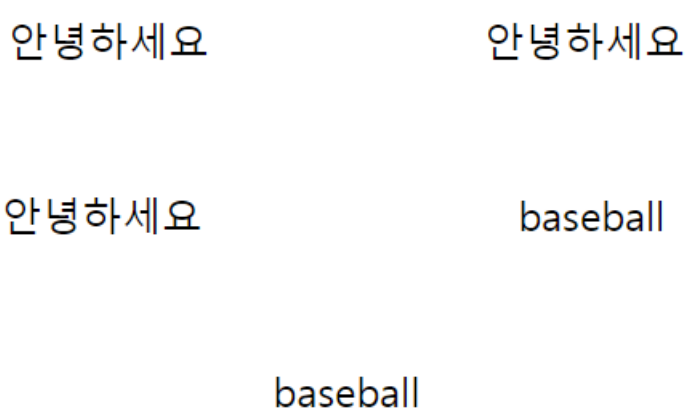
안녕하세요

baseball baseball

CSS Display: Inline-block

- 부모 Element의 text-align:center로 가운데 정렬

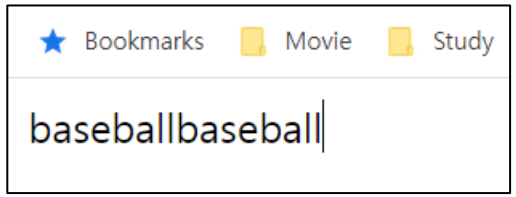
```
<div class="parent">
    <p>안녕하세요</p><p>안녕하세요</p><p>안녕하세요</p>
    <span>baseball</span><span>baseball</span>
</div>

p, span {
    display: inline-block;
    margin: 10px;
    padding: 10px;
    width: 150px;
    height: 30px

}

.parent {
    text-align: center;
}
```



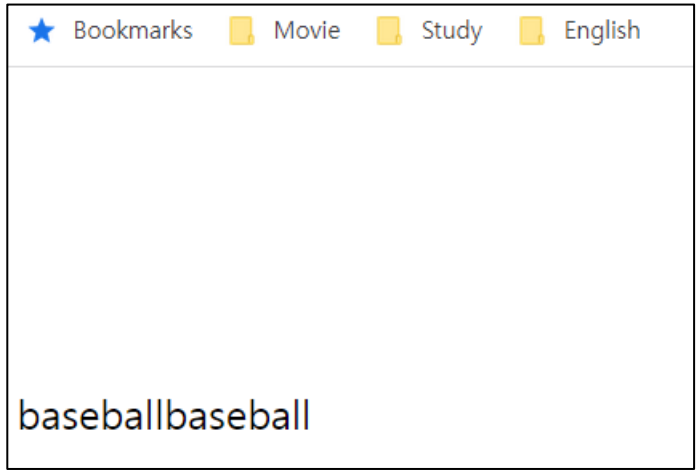
CSS Display: Inline-block

- none: 해당 Element를 화면에서 보이지 않게 함

```
<p>안녕하세요</p><p>안녕하세요</p><p>안녕하세요</p>
<span>baseball</span><span>baseball</span>

p {
    display: none;
}
```



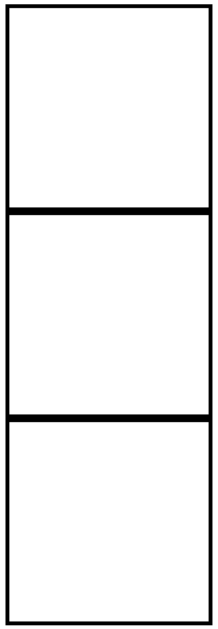
- visibility: hidden과의 차이:
  - display: none은 공간을 유지 하지 않음



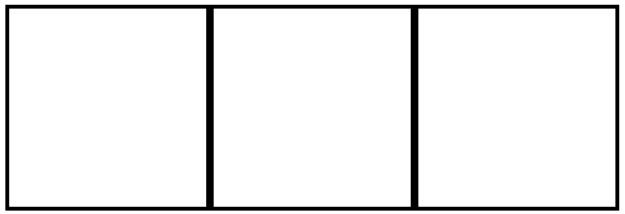
CSS Display: Flexible 박스

- 유연하게 조정한다는 뜻의 새로운 display 속성
- HTML5에서 추가
- 모든 모던 브라우저에서 지원

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



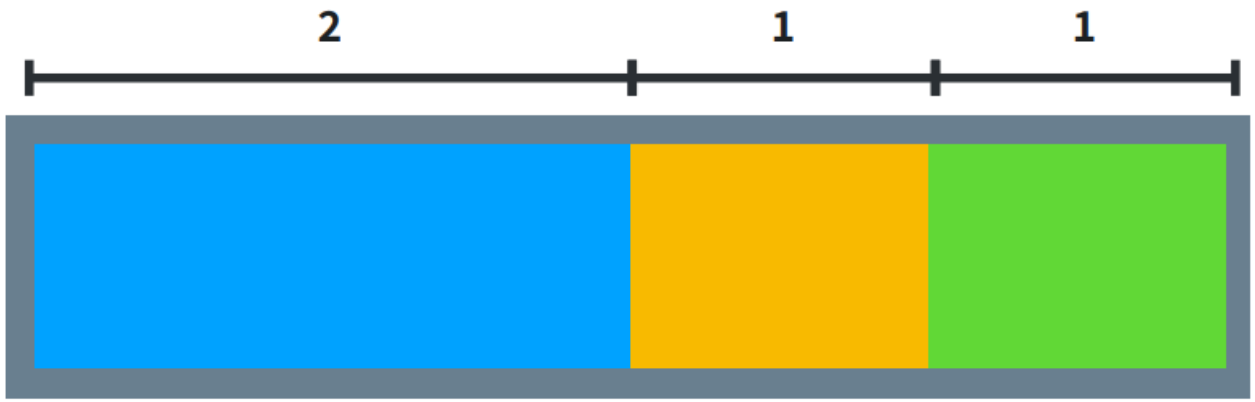
```
.container {
    display: flex;
    /* 생략 가능 */
    flex-direction: row;
}
```



CSS Display: Flexible 박스

- flex: {숫자}: 숫자 비율만큼 공간 차지
- 공간을 배치한 후에 남는 공간을 유동적으로 조절 가능

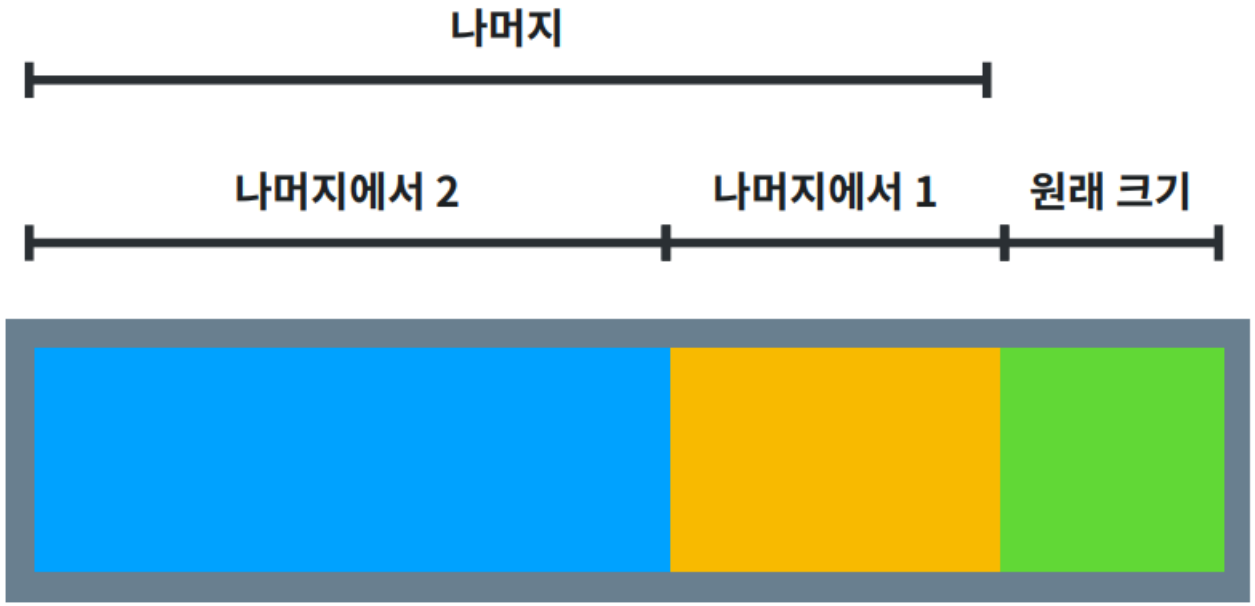
```
.child:nth-child(1) {
    flex: 2;
}

.child:nth-child(2) {
    flex: 1;
}

.child:nth-child(3) {
    flex: 1;
}
```



CSS Display: Flexible 박스

- flex: none : 원래 지정된 공간을 차지
- flex: {숫자}: 나머지 공간을 다시 비율로 나눠서 차지

```
.child:nth-child(1) {
    flex: 2;
}

.child:nth-child(2) {
    flex: 1;
}

.child:nth-child(3) {
    flex: none;
}
```

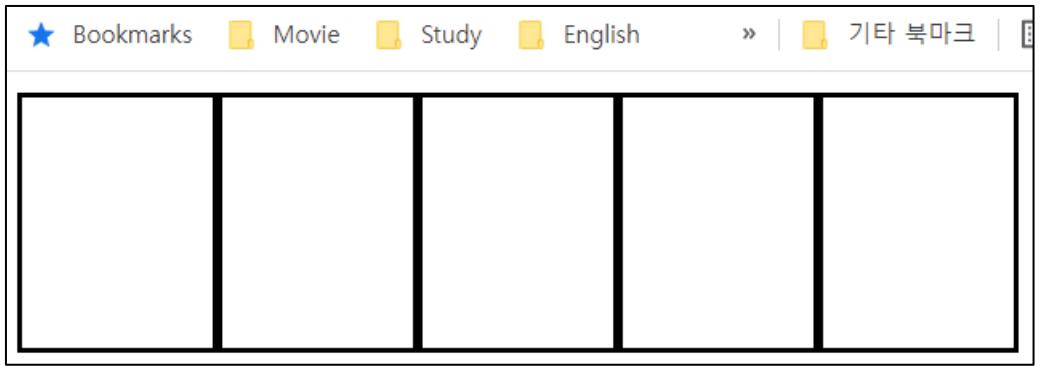


CSS Display: Flexible 박스

- flex wrap: 한 줄을 넘어서도 flex-box가 유지될 수 있도록 함

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

```
.child {
    width: 100px;
    height: 100px;
    border: solid;
}

.container {
    display: flex;
    width: 400px;
}
```

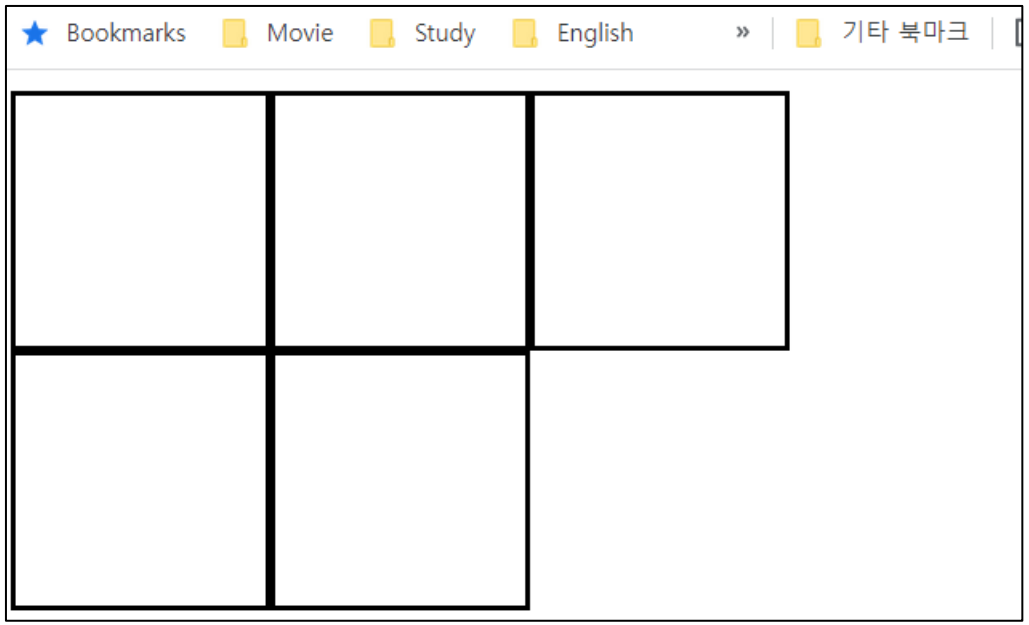


CSS Display: Flexible 박스

- flex wrap: 한 줄을 넘어서도 flex-box가 유지될 수 있도록 함

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

```
.child {
    width: 100px;
    height: 100px;
    border: solid;
}

.container {
    display: flex;
    flex-wrap: wrap;
    width: 400px;
}
```

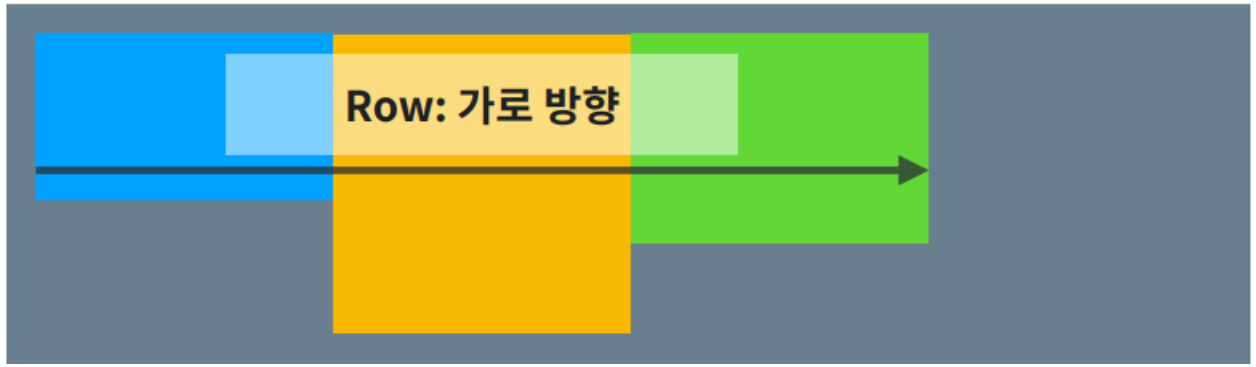


Flex: 정렬 (가로 방향)

- flex: justify-content
  - Flex-direction과 같은 방향을 정렬

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

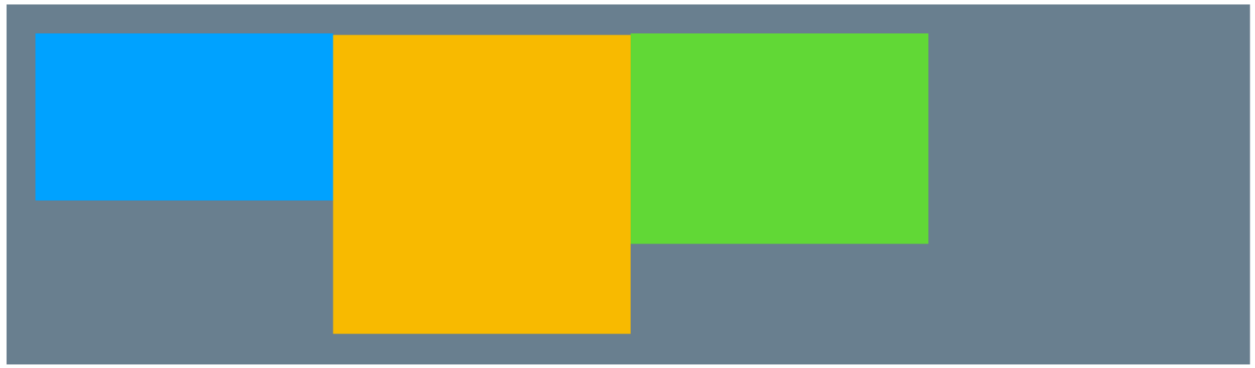
```
.container {
    display: flex;
}
```



Flex: 정렬 (가로 방향)

- flex: justify-content: flex-start
  - 흐름 방향의 시작에서 정렬

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



```
.container {
    display: flex;
    justify-content: flex-start;
}
```

Flex: 정렬 (가로 방향)

- flex: justify-content: center
  - 흐름 방향의 가운데에서 정렬

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

```
.container {
    display: flex;
    justify-content: center;
}
```

Flex: 정렬 (가로 방향)

- flex: justify-content: flex-end
  - 흐름 방향의 끝에서 정렬

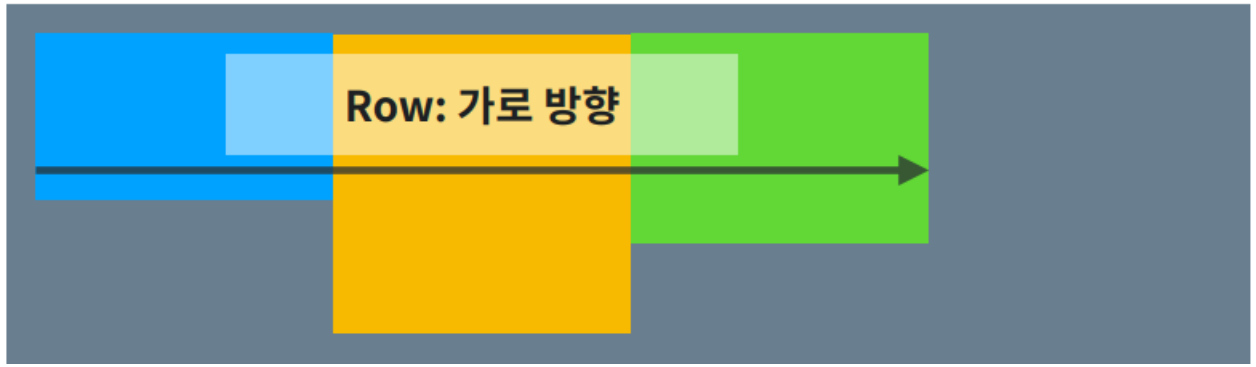
```
<div class="container">
    <div class="child"></div>
    <div class="child"></div>
    <div class="child"></div>
</div>
```

```
.container {
    display: flex;
    justify-content: flex-end;
}
```

Flex: 정렬 (수직 방향)

- flex: align-items
  - flex-direction과 수직 방향을 정렬

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


.container {
    display: flex;
}
```

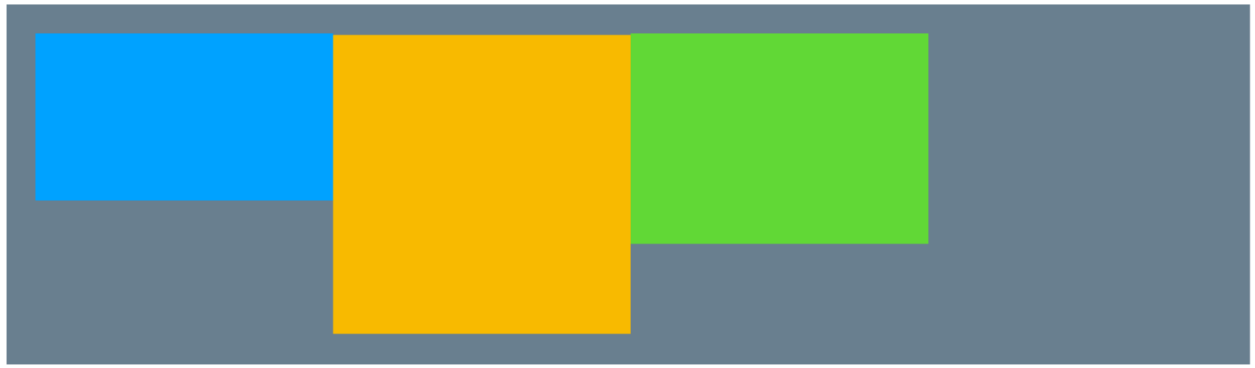


Flex: 정렬 (수직 방향)

- Flex: align-items: flex-start
  - 흐름 수직 방향의 시작에서 수직 정렬

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

```
.container {
    display: flex;
    align-items: flex-start;
}
```



Flex: 정렬 (수직 방향)

- flex: align-items: center
  - 흐름 수직 방향의 가운데에서 수직 정렬

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

```
.container {
    display: flex;
    align-items: center;
}
```

Flex: 정렬 (수직 방향)

- flex: align-items: flex-end
  - 흐름 수직 방향의 끝에서 수직 정렬

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

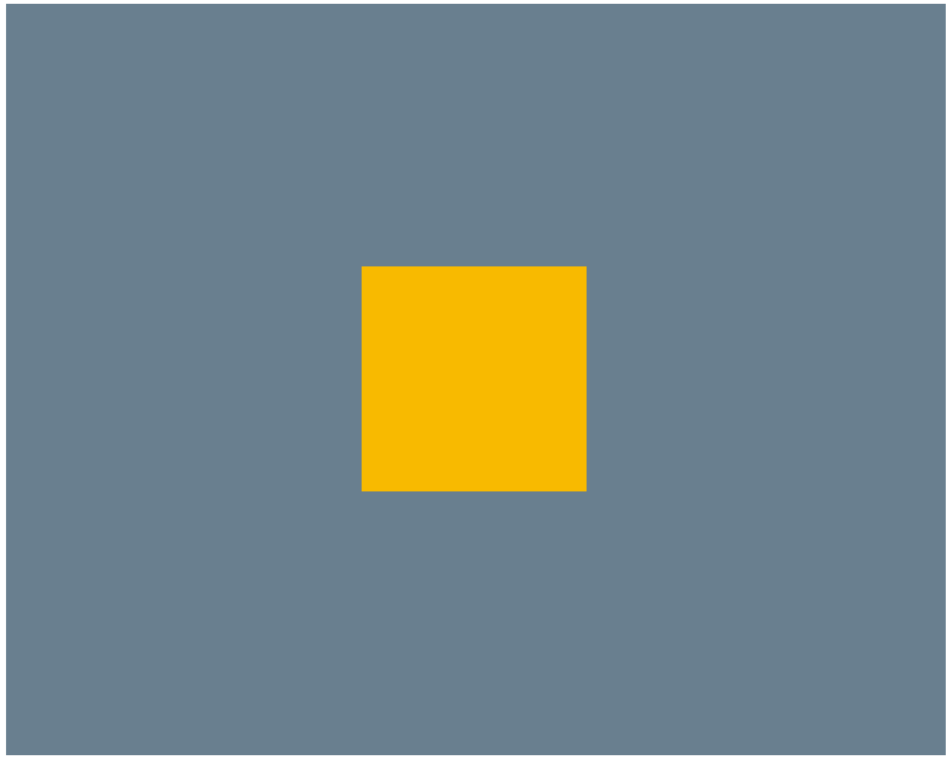
```
.container {
    display: flex;
    align-items: flex-end;
}
```

Flex: 정렬 (가운데 배치)

- 가로, 세로 가운데 배치

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

```
.container {
    display: flex;
    align-items: center;
    justify-content: center
}
```



Position

Element를 배치하는 방법을 지정

Position

- static: 기본 position 값
    - left, right, top, bottom 값 동작 안함

```
<div class="container">
    <div></div>
    <div class="target"></div>
    <div></div>
</div>
```

```
.container {
    display: flex;
    margin: 30px;
    padding: 10px;
}
```

```
.target {
    position: static;
    top: 10px;
    left: 10px;
}
```

Position

- relative : 원래 위치를 기준으로 이동

```
<div class="container">
    <div></div>
    <div class="target"></div>
    <div></div>
</div>
```

```
.container {
    display: flex;
    margin: 30px;
    padding: 10px;
}
```

```
.target {
    position: relative;
    top: 10px;
    left: 10px;
}
```

Position

- absolute : position: relative를 가진 가장 가까운 상위 Element를 기준으로 위치
  - 없으면 body를 기준으로 위치

```
<div class="container">
    <div></div>
    <div class="target"></div>
    <div></div>
</div>
```

```
.container {
    display: flex;
    margin: 30px;
    padding: 10px;
    position: relative;
}
.target {
    position: absolute;
    top: 10px;
    left: 10px;
}
```

Position

- flxed : transform 속성값을 가진 가장 가까운 상위 Element를 기준으로 위치
  - 없으면 화면을 기준으로 위치

```
<div class="container">
    <div></div>
    <div class=“target"></div>
    <div></div>
</div>
```

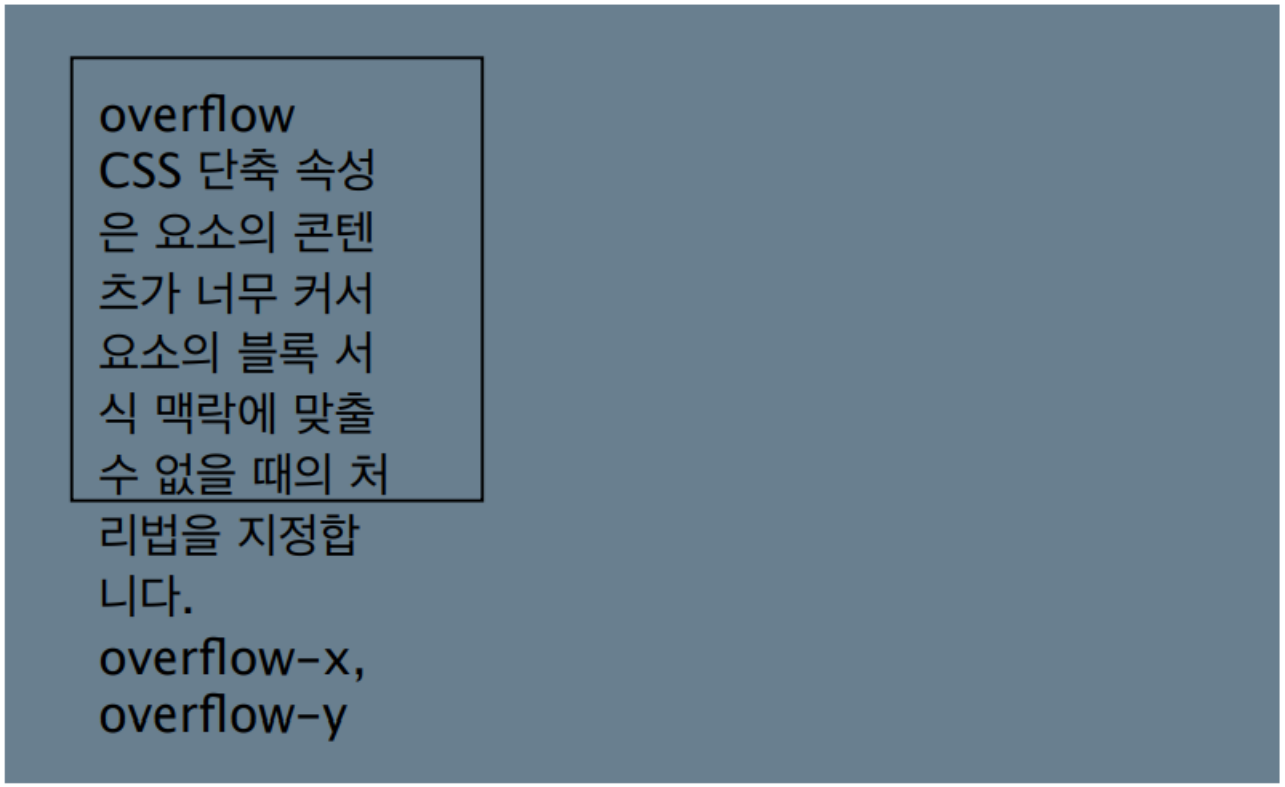
```
.container {
    display: flex;
    margin: 30px;
    padding: 10px;
    position: relative;
}
.target {
    position: fixed;
    top: 10px;
    left: 10px;
}
```

Overflow

Overflow

- 자식 Element가 부모 Element를 넘어 갈 때 렌더링 방식

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


.container {
    padding: 10px;
    outline: solid black;
    width: 100px;
    height: 100px;
}

box {
  width: 80px;
  height: 200px;
}
```

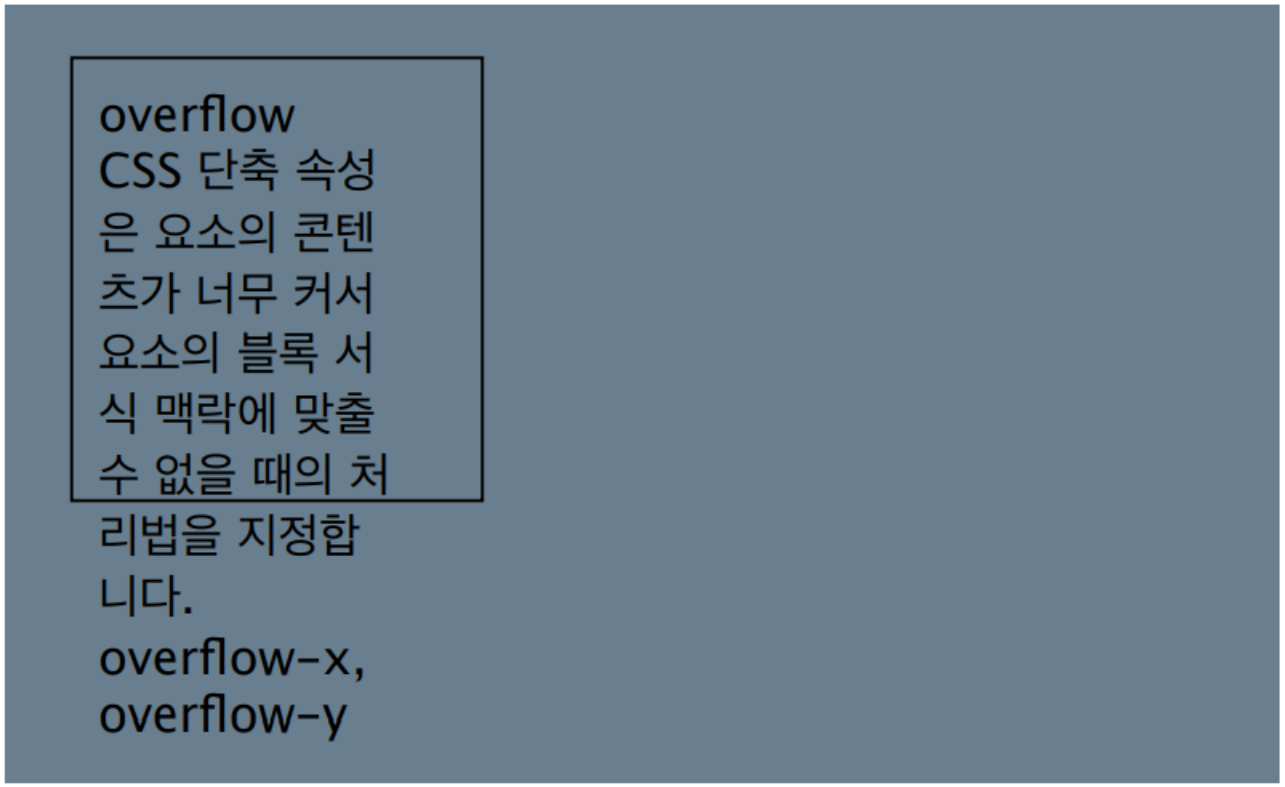


Overflow: visible

- visible: 내용을 자르지 않고 그대로 보여 줌 (기본 값)

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

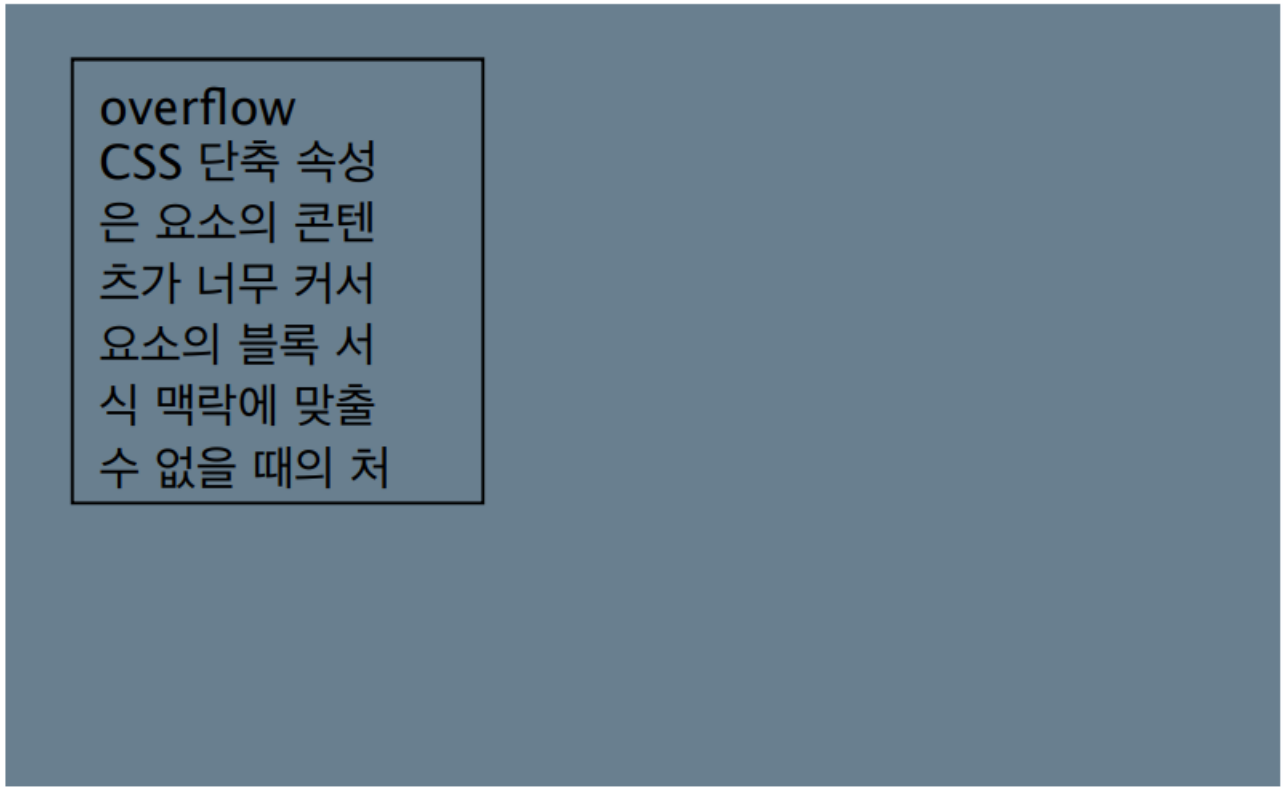
```
.container {
    padding: 10px;
    outline: solid black;
    width: 100px;
    height: 100px;
    overflow: visible;
}

box {
  width: 80px;
  height: 200px;
}
```



Overflow: hidden

- hidden: 부모 Element를 넘어 가는 값을 잘라 냄

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


.container {
    padding: 10px;
    outline: solid black;
    width: 100px;
    height: 100px;
    overflow: hidden;
}

box {
  width: 80px;
  height: 200px;
}
```



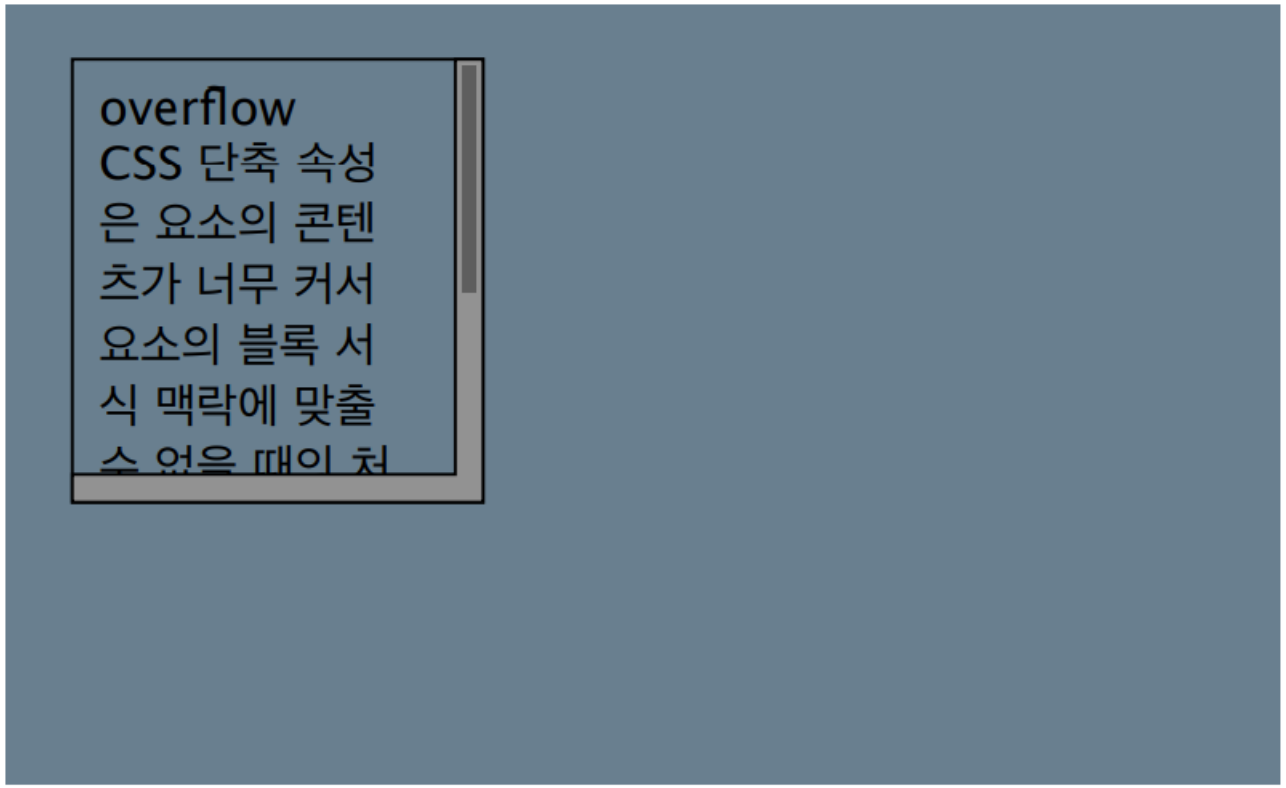
Overflow: scroll

- scroll: 가로 세로 스크롤 바를 항상 보여줌

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

```
.container {
    padding: 10px;
    outline: solid black;
    width: 100px;
    height: 100px;
    overflow: scroll;
}
```

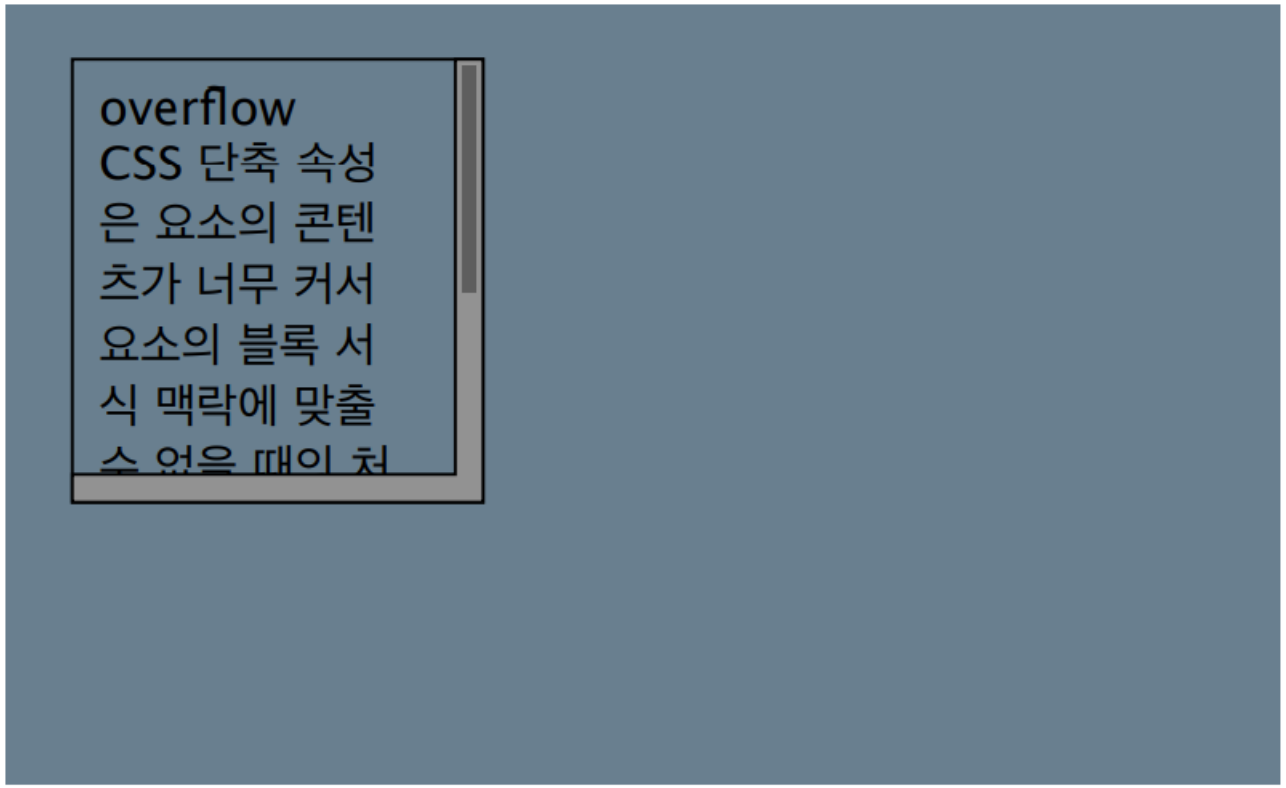
```
box {
  width: 80px;
  height: 200px;
}
```



Overflow: auto

- auto: 내용이 넘치는 경우에만 스크롤바를 보여 줌

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


.container {
    padding: 10px;
    outline: solid black;
    width: 100px;
    height: 100px;
    overflow: auto;
}

box {
  width: 80px;
  height: 200px;
}
```



Overflow: text-overflow

- 텍스트가 부모 요소를 벗어났을 때 말 줄임 표시를 위해 사용

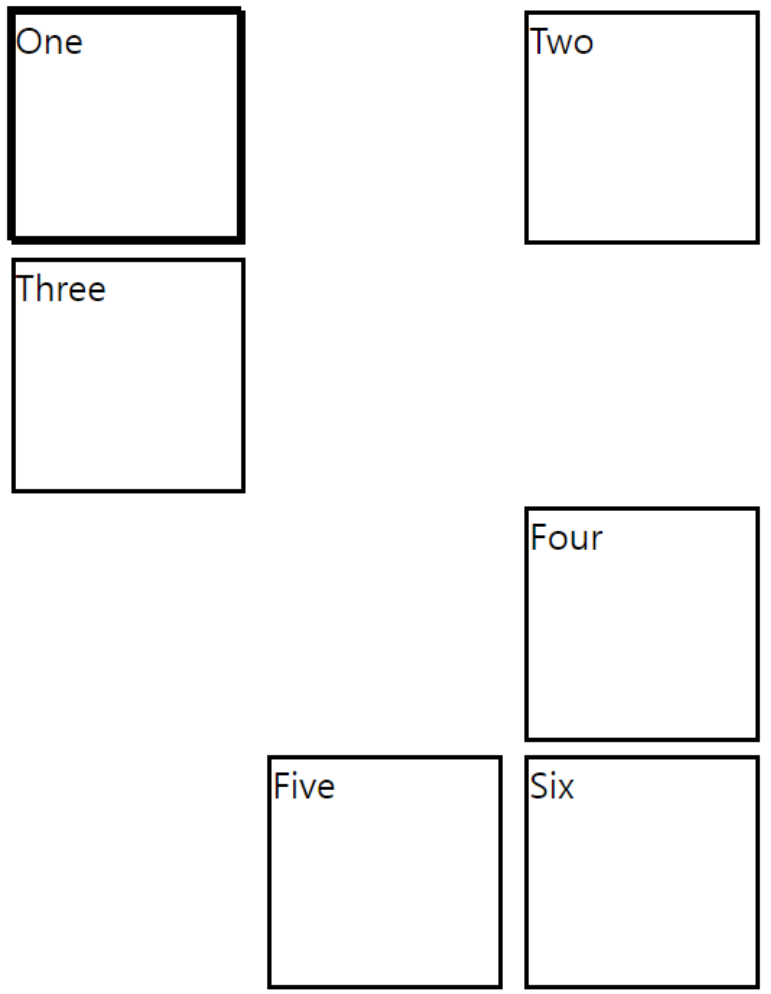
```
<div class="container">
    <div class="box">The text-overflow sets how
hidden overflow content is signaled to users
</div>
</div>


.box {
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
}
```

CSS Display: grid

- 새로운 Display 속성
- 레이아웃을 더 정교하게 나누거나 계층을 잘 컨트롤 할 수 있도록 함

```
<div class="wrapper">
    <div class="one">One</div>
    <div class="two">Two</div>
    <div class="three">Three</div>
    <div class="four">Four</div>
    <div class="five">Five</div>
    <div class="six">Six</div>
</div>
```

```
.wrapper {
 display: grid;
 grid-template-columns: repeat(3, 1fr);
 gap: 10px;
 grid-auto-rows: minmax(100px, auto);
}
.one {
 grid-column: 1 / 3;
 grid-row: 1;
}
.two {
 grid-column: 3 / 3;
 grid-row: 1 / 3;
}
.three {
 grid-column: 1;
 grid-row: 2 / 5;
}
.four {
 grid-column: 3;
 grid-row: 3;
}
.five {
 grid-column: 2;
 grid-row: 4;
}
.six {
 grid-column: 3;
 grid-row: 4;
}
```

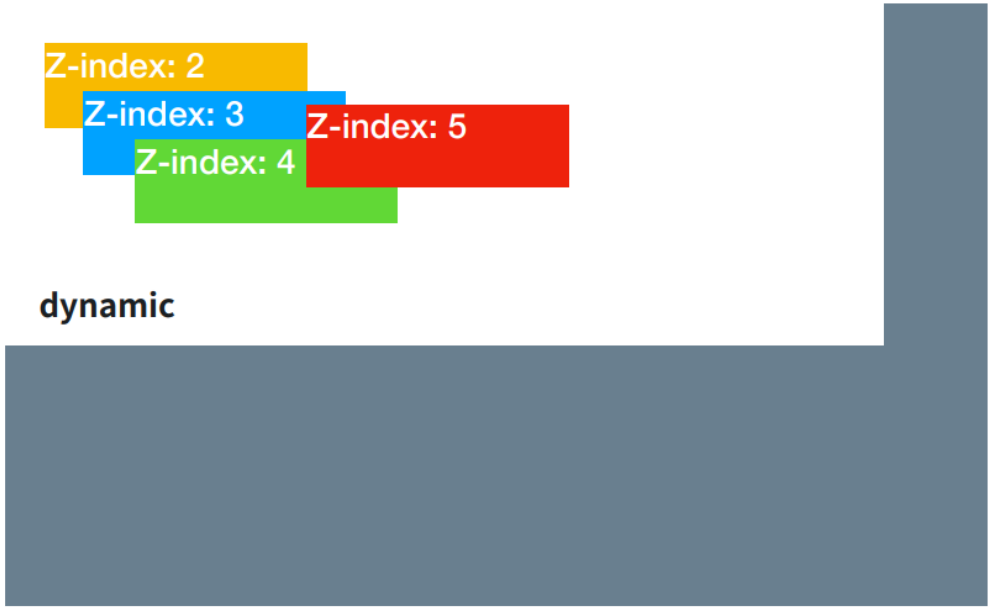


Z-index

z-index

- 어떤 Element가 가장 위로 올라갈 것인지 결정
- position 속성이 static이 아닌 다른 값으로 설정되어야 함

```
<div class="box dynamic">dynamic</div>
<div class="box box2">z-index: 2</div>
<div class="box box3">z-index: 3</div>
<div class="box box4">z-index: 4</div>
<div class="box box5">z-index: 5</div>

.box {
  position: absolute;
}
```

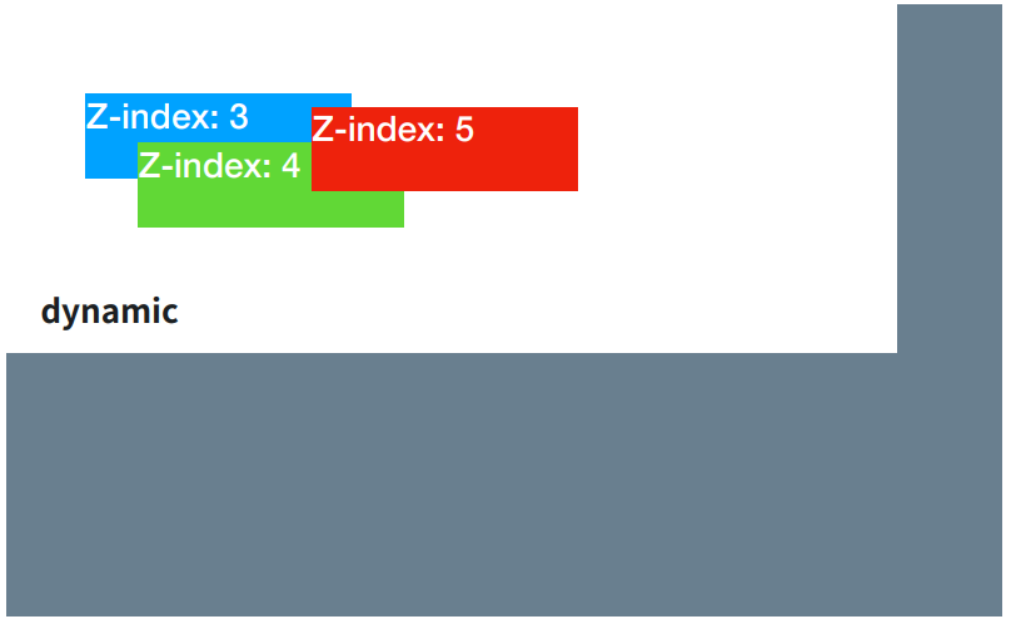


z-index

- z-index를 이용하여 요소의 순서 설정

```
<div class="box dynamic">dynamic</div>
<div class="box box2">z-index: 2</div>
<div class="box box3">z-index: 3</div>
<div class="box box4">z-index: 4</div>
<div class="box box5">z-index: 5</div>
```

```
.box {
  position: absolute;
}

.box2 {
  z-index: 1;
}

.dynamic {
  z-index: 2;
}
```



글자와 아이콘

Font 적용

- 글자에 적용할 수 있는 속성

속성	설명	값
font-style	글자 스타일	Normal | italic | oblique
font-weight	글자 굵기	Lighter | normal | bold | bolder | 1 ~ 100
font-size	글자 크기	<숫자><단위>
font-family	글꼴	글꼴 이름
line-height	기본 글꼴의 상대적인 크기	normal | <숫자> | <숫자><단위>

- 글자 크기 표현 단위

단위	설명
px	화소 단위
em	부모 요소의 글자 크기 기준 배율
rem	HTML 글가 크기 기준 배율
pt	글꼴에 많이 쓰임
%	기본 글꼴의 상대적인 크기
vw, vh	뷰포트 기준 너비, 높이

Font 적용

- 필요한 속성을 하나씩 적용

```
div {
  font-style: italic;
  font-weight: bold;
  font-size: .8em;
  line-height: 1.2;
  font-family: Arial, Helvetica, sans-serif;
}
```

- font 속성으로 한번에 적용

```
div {
  font: italic bold .8em/1.2 Arial, Helvetica,
sans-serif;
}
```

Web font

- 클라이언트 컴퓨터에 글꼴이 없을 경우 웹 폰트를 통해 다운로드 하게 할 수 있음
- 외부 font resource를 위한 링크 태그 또는 css의 import를 통해 받을 수 있음

클라이언트의 컴퓨터에 설치되지 않은 폰트를 사이트에서 사용

1. 적용할 폰트의 경로와 이름 확인

2. 적용할 경로와 이름 선언

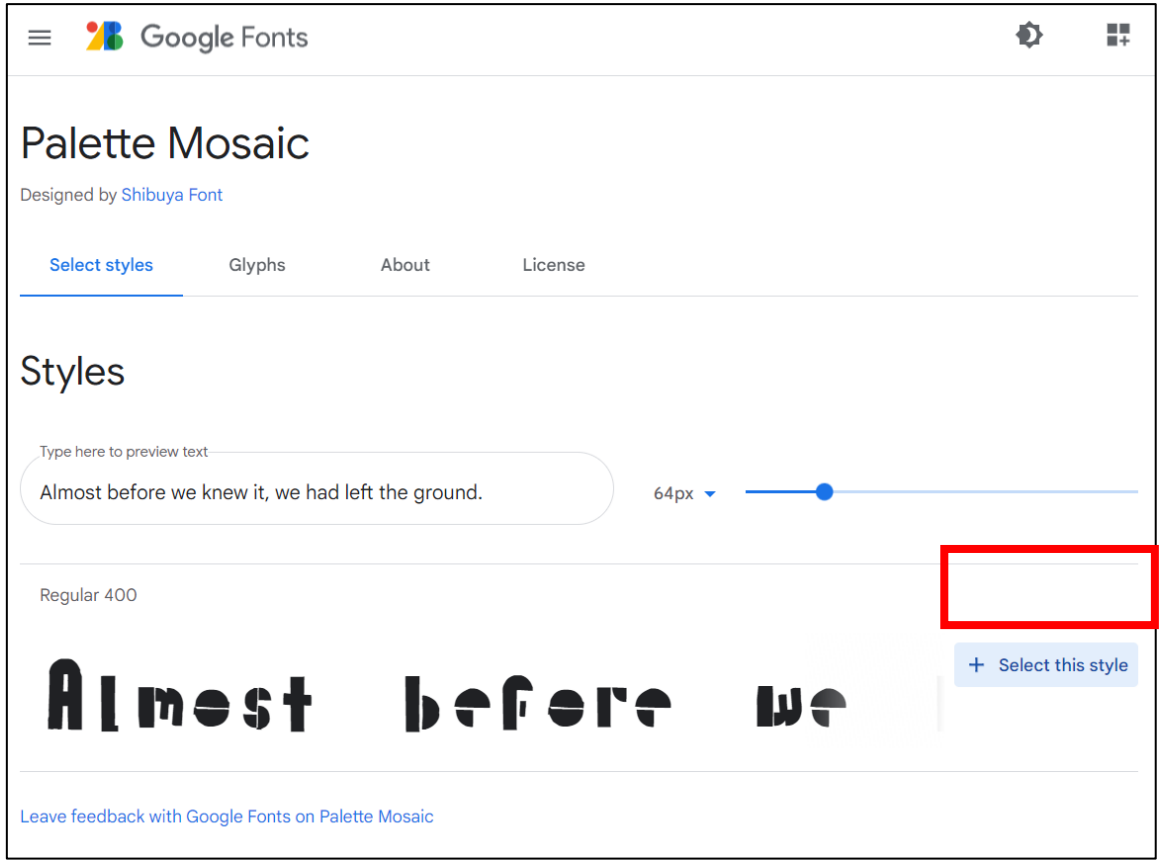
```
@font-face {
  font-family: "American Captain";
  src: url(Captain.oft);
}
```

3. 폰트 사용

```
<div class="box">A computer is like a
violin.</div>

div {
  font-family: American Captain;
  font-size: 20px;
}
```

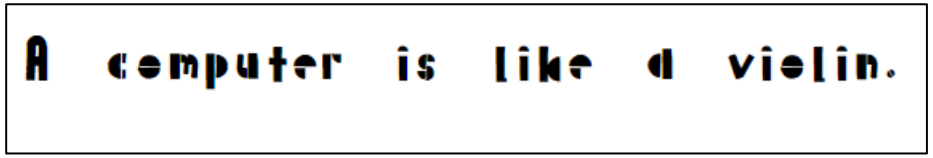
Web font – Google font

- https://fonts.google.com에서 웹 폰트 사용 가능
- 링크 태그 또는 css import를 통해 사용



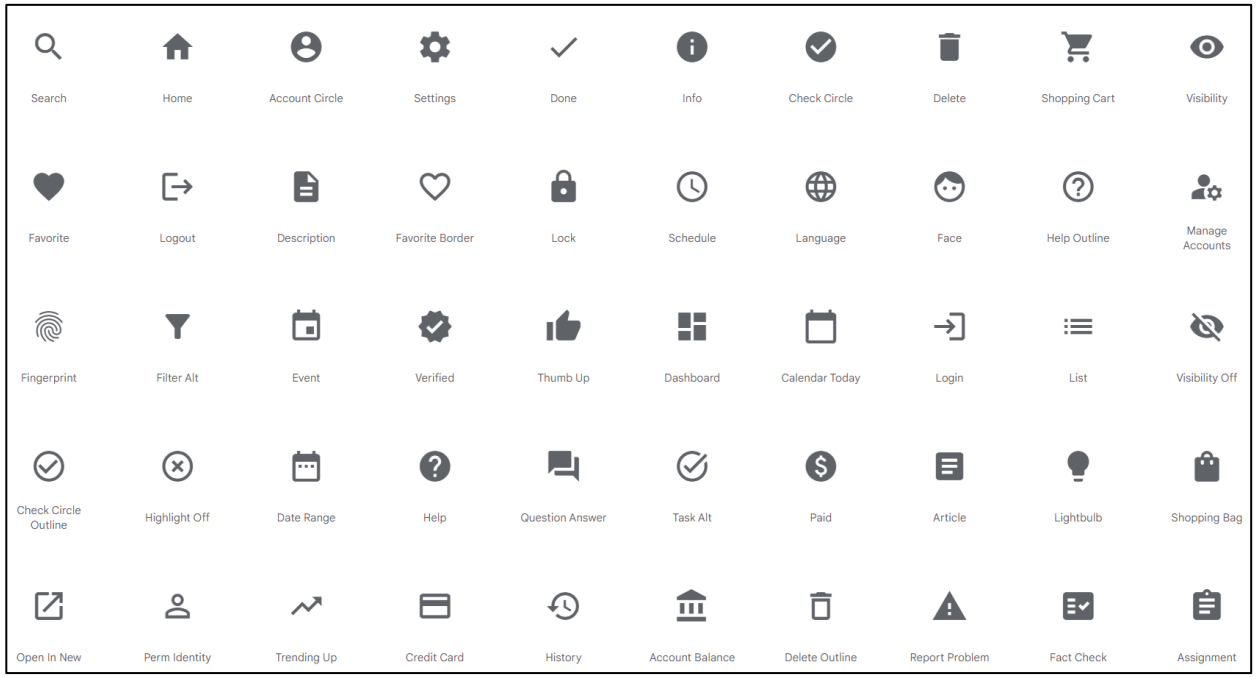
Web font – Google font

```
@import
url('https://fonts.googleapis.com/css2?family=Palette+Mosaic&display=swap');

div {
  font-family: 'Palette Mosaic', cursive;
  font-size: 20px;
}
```



Web font 아이콘 활용

- Meterial-icon 사용
- https://google.github.io/material-design-icons/
- https://fonts.google.com/icons?selected=Material+Icons
- 벡터 방식으로 아이콘이 깨지지 않음
- 폰트이므로 크기와 색상 조절 가능
- CSS 파일만 import하면
  간단한 html 태그로 사용



Material Icon

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

<body>
  <span class="material-icons" style="font-size:30pt;">face</span>
  <span class="material-icons" style="color:green;">search</span>
</body>
```