

# HTML5 & CSS3

---

Client Side Language

장민창

mcjang@hucloud.co.kr

# HTML5

---

1. HTML5 문서 구조
2. 태그의 종류
3. Attributes
4. Events

# 1. HTML5 문서 구조

# 1. HTML5 문서 구조

- HTML (HyperText Markup Language)
  - 태그(Node / Element)를 조합해 하나의 문서를 만드는 언어.
  - HTML로 만든 파일은 브라우저로 결과를 즉시 확인 해 볼 수 있다.
    - Client Side Language (HTML, CSS, JS 등)
- HTML의 구조
  - 문서 타입
  - html
  - head
    - title
    - meta
    - script
    - style
  - body
    - tags

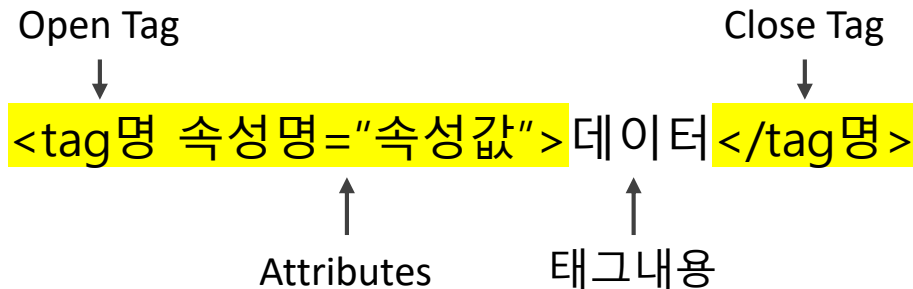


# 1. HTML5 문서 구조

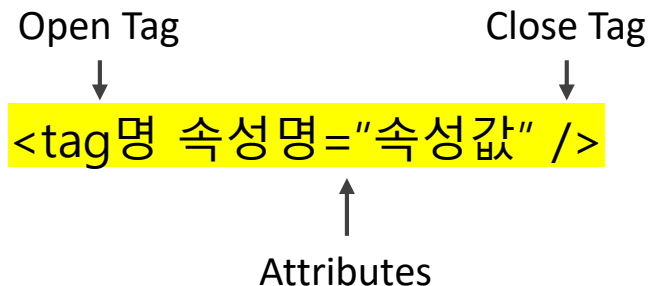
- 태그 (Tag)
  - Element, Node 라고도 표현.
  - HTML은 다양한 형태의 데이터를 표현하기 위해 많은 태그를 미리 정의해 두었음.

- 태그를 사용하는 방법

- 태그는 Open Tag, Attributes, 태그내용, Close Tag로 구성되어 있음.



- 태그에 따라 Open Tag와 Close Tag가 하나로 구성된 경우도 있음. (Single ling tag)



# 1. HTML5 문서 구조

---

- 태그 내용

- 간단한 텍스트 데이터

```
<tag명 속성명="속성값">텍스트</tag명>
```

- 복잡한 레이아웃을 만들기 위해 다른 태그를 사용할 수 도 있음.

```
<tag명 속성명="속성값">  
  <tag명>  
    <tag명>  
    </tag명>  
  </tag명>  
</tag명>
```

## 2. 태그의 종류

## 2. 태그의 종류

- W3Schools HTML Element Reference (<https://www.w3schools.com/tags/default.asp>)
  - HTML의 모든 Tag 목록을 정리해 둔 사이트.

### HTML Element Reference

[< Home](#)[Next >](#)

#### HTML Tags Ordered Alphabetically

Tag	Description
<a href="#"><u>&lt;!--...--&gt;</u></a>	Defines a comment
<a href="#"><u>&lt;!DOCTYPE&gt;</u></a>	Defines the document type
<a href="#"><u>&lt;a&gt;</u></a>	Defines a hyperlink
<a href="#"><u>&lt;abbr&gt;</u></a>	Defines an abbreviation or an acronym
<a href="#"><u>&lt;acronym&gt;</u></a>	Not supported in HTML5. Use <a href="#"><u>&lt;abbr&gt;</u></a> instead. Defines an acronym
<a href="#"><u>&lt;address&gt;</u></a>	Defines contact information for the author/owner of a document
<a href="#"><u>&lt;applet&gt;</u></a>	Not supported in HTML5. Use <a href="#"><u>&lt;embed&gt;</u></a> or <a href="#"><u>&lt;object&gt;</u></a>



## 2. 태그의 종류

- HTML의 주요 태그

### HTML 문서와 관련된 태그

태그명	부모태그	설명
<!DOCTYPE html>	-	HTML 문서 타입 정의 (HTML5 사용시 필수)
<html>	-	HTML 문서의 최상단 태그 (필수)
<head>	<html>	HTML 문서의 정보를 설정 (옵션) Javascript, CSS 적용 시 필수
<script>	<head>	HTML 문서에서 Javascript 적용시 설정 여는 태그와 닫는 태그 사이에 Javascript를 작성
<title>	<head>	HTML 문서의 제목 여는 태그와 닫는 태그 사이에 문서의 제목을 작성. 브라우저 탭에 문서의 제목이 표시된다.
<meta>	<head>	HTML 문서의 메타데이터(부가정보)를 작성한다.
<base>	<head>	문서에서 사용할 모든 링크의 상대 경로의 절대 주소를 설정
<style>	<head>	HTML 문서의 CSS 설정. 여는 태그와 닫는 태그 사이에 Internal Style을 작성한다.
<link>	<head>	HTML 문서의 CSS 설정. 외부 CSS의 파일을 가져와 문서에 적용한다 (External Style)
<body>	<html>	HTML 문서 본문의 최상단 태그 (필수) 태그와 태그 사이에 본문 내용을 작성)
<!-- 주석 내용 -->	-	HTML 문서에 주석을 작성.

## 2. 태그의 종류

- HTML의 주요 태그

표와 관련된 태그

태그명	부모태그	설명
<table>	<body> 내부	테이블(표)을 정의함
<caption>	<table>	테이블의 설명(제목)을 정의함
<colgroup>	<table>	테이블의 컬럼(열)의 너비를 정의함
<col>	<colgroup>	테이블의 컬럼(열)의 너비를 지정함
<thead>	<table>	테이블의 타이틀을 정의함
<tbody>	<table>	테이블의 콘텐츠를 정의함
<tr>	<table>, <thead>, <tbody>	테이블에 행 하나를 생성함
<th>	<tr>	테이블에 타이틀 열 하나를 생성함
<td>	<tr>	테이블에 콘텐츠 열 하나를 생성함

## 2. 태그의 종류

- HTML의 주요 태그

### Form과 관련된 태그

태그명	부모태그	설명
<form>	<body> 내부	서버로 데이터를 보내기 위한 양식을 정의함
<input>	<form>	사용자가 입력할 수 있는 필드를 정의함 입력 값에 따라 많은 type이 제공된다.
<label>	<form>	input이나, textarea, select 등의 제목을 정의함. Label을 클릭하면 관련된 form 태그가 포커싱 된다.
<button>	<form>	클릭할 수 있는 버튼을 정의함
<textarea>	<form>	여러 줄의 텍스트를 입력할 수 있는 필드를 정의함.
<select>	<form>	드롭다운박스를 정의함
<optgroup>	<select>	드롭다운박스의 아이템 그룹을 정의함
<datalist>	<form>	input 태그의 자동완성 아이템을 정의함
<option>	<select>, <optgroup>, <datalist>	드롭다운박스의 아이템을 정의함

## 2. 태그의 종류

- HTML의 주요 태그

멀티미디어와 관련된 태그

태그명	부모태그	설명
<audio>	<body> 내부	음성 파일을 재생할 수 있는 컨트롤을 정의함
<video>	<body> 내부	영상 파일을 재생할 수 있는 컨트롤을 정의함
<source>	<audio>, <video>	음성, 영상 파일의 위치(URL)을 정의함
<img>	<body> 내부, <figure>	이미지 파일을 보여줌.
<map>	<body> 내부	이미지 파일에 영역을 만들어 클릭할 수 있도록 정의함.
<area>	<map>	이미지 파일에 영역을 만들어 클릭할 수 있도록 정의함.
<figure>	<body> 내부	사진과 설명이 함께 필요할 때 사용
<figcaption>	<figure>	사진의 설명을 정의

## 2. 태그의 종류

- HTML의 주요 태그

### 목록과 관련된 태그

태그명	부모태그	설명
<ol>	<body> 내부	Ordered List. 1, 2, 3 등의 숫자로 문서를 작성
<ul>	<body> 내부	Unordered List. ● 등의 기호로 문서를 작성
<li>	<ol>, <ul>	List의 Item을 작성
<dl>	<body> 내부	문서 목록을 정의
<dt>	<dl>	문서 챕터의 이름을 정의
<dd>	<dl>	문서 챕터 내용을 정의

### 영역과 관련된 태그

태그명	부모태그	설명
<div>	<body> 내부	문서의 영역을 지정. 시멘틱 태그가 나오기 전에 주로 사용.
<details>	<body> 내부	내용 줄이기 영역을 지정
<summary>	<details>	줄여진 내용을 작성
<dialog>	<body> 내부	레이어 팝업을 정의
<iframe>	<body> 내부	외부 문서를 페이지 영역 일부에 별도로 보여주도록 정의.
<template>	<body> 내부	자주 사용되는 태그 목록을 별도로 지정해놓고 Javascript로 제어

## 2. 태그의 종류

- HTML의 주요 태그

텍스트와 관련된 태그

태그명	부모태그	설명
<a>	<body> 내부	다른 문서로 이동시키는 Anchor를 정의 (링크)
<b>	<body> 내부	문서를 굵게 표현함
 	<body> 내부	강제 개행함.
<h1> to <h6>	<body> 내부	<h1> 부터 <h6> 까지 있음., Headline을 정의
<i>	<body> 내부	텍스트를 기울임. 강조하고 싶은 텍스트에 정의
<mark>	<body> 내부	텍스트 배경색에 하이라이트 표시. 강조하고 싶은 텍스트에 정의
<p>	<body> 내부	문단을 정의함.
<pre>	<body> 내부	태그 내에 정의된 텍스트를 원본 그대로 보여줌 (태그 제외)
<span>	<body> 내부	일부 텍스트에 스타일을 정의하기 위해 사용.

## 2. 태그의 종류

- HTML의 주요 태그

시멘틱 웹과 관련된 태그

태그명	부모태그	설명
<header>	<body> 내부	HTML 문서의 최상단 혹은 각 영역의 헤더 부분을 정의
<nav>	<body> 내부	다른 페이지로 이동할 수 있는 링크 집합을 정의
<section>	<body> 내부	HTML 문서의 각 영역을 정의
<article>	<body> 내부	HTML 문서 내에 관련된 데이터를 보여주도록 정의
<aside>	<body> 내부	HTML 본문 옆(Side bar) 영역을 정의.
<footer>	<body> 내부	HTML 문서의 최하단 영역을 정의
<main>	<body> 내부	HTML 문서의 메인 콘텐츠를 정의

# 3. Attributes



# 3. Attributes

- HTML의 주요 공통 Attribute

속성명	설명
accesskey	단축키로 엘리먼트를 활성화 하거나 포커싱 할 수 있습니다. (Alt + 단축키로 사용합니다.)
class	엘리먼트에 CSS를 위한 클래스를 정의할 수 있습니다.
data-*	엘리먼트에 사용자 정의 변수를 정의할 수 있습니다.
hidden	엘리먼트를 숨길 수 있습니다
id	엘리먼트에 CSS를 위한 ID를 정의할 수 있습니다. ID 값은 고유해야 합니다.
style	엘리먼트에 Inline CSS를 정의할 수 있습니다.
tabindex	탭을 눌러 포커싱 하기 위한 순서를 지정합니다.
title	엘리먼트에 마우스를 올렸을 때, 툴팁이 나타나게 합니다. 사용자 접근성을 위해 반드시 필요합니다.

Open Tag



Close Tag



`<tag명 속성명="속성값" 속성명="속성값">데이터</tag명>`



Attributes



태그내용

# 3. Attributes

- HTML 태그의 Attribute
  - 각 태그들의 Attribute는 아래 URL에서 확인한다.
  - [https://www.w3schools.com/tags/tag\\_태그명.asp](https://www.w3schools.com/tags/tag_태그명.asp)
    - 예> link 태그의 Attribute 확인
  - [https://www.w3schools.com/tags/tag\\_link.asp](https://www.w3schools.com/tags/tag_link.asp)

## Attributes

Attribute	Value	Description
<u>crossorigin</u>	anonymous use-credentials	Specifies how the element handles cross-origin requests
<u>href</u>	URL	Specifies the location of the linked document
<u>hreflang</u>	language_code	Specifies the language of the text in the linked document
<u>media</u>	media_query	Specifies on what device the linked document will be displayed
<u>referrerpolicy</u>	no-referrer no-referrer-when-downgrade origin origin-when-cross-origin unsafe-url	Specifies which referrer to use when fetching the resource
<u>rel</u>	alternate author dns-prefetch help icon license next pingback preconnect prefetch preload prerender prev search stylesheet	Required. Specifies the relationship between the current document and the linked document
<u>sizes</u>	HeightxWidth any	Specifies the size of the linked resource. Only for rel="icon"
<u>title</u>		Defines a preferred or an alternate stylesheet
<u>type</u>	media_type	Specifies the media type of the linked document

## 4. Events

# 4. Events

- HTML의 주요 공통 Event

## Form Events

이벤트 속성명	값	설명
onblur	script	엘리먼트가 포커스를 잃었을 때 스크립트 실행
onchange	script	엘리먼트의 내용이 변경되었을 때 스크립트 실행
oncontextmenu	script	엘리먼트에 오른쪽 클릭했을 때 스크립트 실행
onfocus	script	엘리먼트에 포커스가 부여되었을 때 스크립트 실행
oninput	script	엘리먼트에 키보드 입력이 발생했을 때 스크립트 실행
oninvalid	script	required 속성이 부여된 엘리먼트의 값 검증을 실패 했을 때 스크립트 실행 (Form submit시 수행)
onreset	script	input type="reset" 버튼을 클릭했을 때 스크립트 실행 (form 태그에 부여)
onsearch	script	input type="search" 에 값 입력 후 엔터를 입력했을 때 스크립트 실행
onsubmit	script	form을 전송할 때 스크립트 실행

Open Tag



<tag명 속성명="속성값" 이벤트명="Javascript Code">데이터</tag명>

Close Tag



Attributes



Events



태그내용



# 4. Events

- HTML의 주요 공통 Event

## Keyboard Events

이벤트 속성명	값	설명
onkeydown	<i>script</i>	엘리먼트에 키를 눌렀을 때 스크립트 실행
onkeypress	<i>script</i>	엘리먼트에 키를 누르고 있을 때 스크립트 실행
onkeyup	<i>script</i>	엘리먼트에 키를 뗐을 때 스크립트 실행

## Mouse Events

이벤트 속성명	값	설명
onclick	<i>script</i>	엘리먼트를 클릭 했을 때 스크립트 실행
ondblclick	<i>script</i>	엘리먼트를 더블 클릭 했을 때 스크립트 실행
onmousedown	<i>script</i>	엘리먼트 위에서 마우스 버튼을 눌렀을 때 스크립트 실행
onmousemove	<i>script</i>	엘리먼트 위에서 마우스를 움직일 때 스크립트 실행
onmouseout	<i>script</i>	엘리먼트에서 마우스 커서가 바깥으로 나갔을 때 스크립트 실행
onmouseover	<i>script</i>	엘리먼트 위에 마우스 커서가 들어왔을 때 스크립트 실행
onmouseup	<i>script</i>	엘리먼트 위에서 마우스 버튼을 뗐을 때 스크립트 실행
onwheel	<i>script</i>	엘리먼트 위에서 마우스 휠 입력 했을 때 스크립트 실행 (스크롤바와 관계 없이 스크롤을 했을 때)
onscroll	<i>script</i>	엘리먼트 위에서 스크롤을 할 때 스크립트 실행 (스크롤바를 움직이는 이벤트)

# 4. Events

- HTML의 주요 공통 Event

## Drag Events

이벤트 속성명	값	설명
ondrag	<i>script</i>	엘리먼트를 드래그 할 때 스크립트 실행
ondragend	<i>script</i>	엘리먼트의 드래그가 종료되었을 때 스크립트 실행
ondragenter	<i>script</i>	드래그 중인 엘리먼트가 드랍영역에 들어왔을 때 스크립트 실행
ondragleave	<i>script</i>	드래그 중인 엘리먼트가 드랍영역에서 나갔을 때 스크립트 실행
ondragover	<i>script</i>	드래그 중인 엘리먼트가 드랍영역 위로 올라왔을 때 스크립트 실행
ondragstart	<i>script</i>	엘리먼트 드래그를 시작할 때 스크립트 실행
ondrop	<i>script</i>	드래그 중인 엘리먼트를 드랍영역에서 놓았을 때 스크립트 실행

## Clipboard Events

이벤트 속성명	값	설명
oncopy	<i>script</i>	엘리먼트의 내용을 복사할 때 스크립트 실행
oncut	<i>script</i>	엘리먼트의 내용을 잘라낼 때 스크립트 실행
onpaste	<i>script</i>	엘리먼트에 복사한 내용을 붙여넣었을 때 스크립트 실행

# 4. Events

- HTML의 주요 공통 Event

## Media Events

이벤트 속성명	값	설명
onabort	script	미디어의 로드가 중단되었을 경우 스크립트 실행
oncanplay	script	미디어의 재생을 시작할 수 있을 때(미디어가 충분히 버퍼를 모았을 때) 스크립트 실행
oncanplaythrough	script	미디어를 버퍼링 중지 없이 재생할 수 있을 때 스크립트 실행
onemptied	script	미디어가 load() 메소드에 의해 초기화 되거나 치명적인 오류가 있어 미디어 리소스를 준비하지 못했을 때
onended	script	미디어 끝에 도달했거나 더 이상 사용할 수 있는 데이터가 없어 재생이 중지됐을 때 스크립트 실행
onerror	script	미디어 리소스 로드에 실패했거나 사용할 수 없을 경우 스크립트 실행
onloadeddata	script	미디어의 현재 재생 위치에 있는 프레임이 로드를 완료했을 때 스크립트 실행
onloadedmetadata	script	미디어의 메타 데이터가 로드가 완료되면 스크립트 실행(영상길이 등)
onloadstart	script	브라우저가 미디어 리소스를 다운로드 시작할 때 스크립트 실행
onpause	script	재생이 일시 중지될 때 스크립트 실행
onplay	script	play() 메소드 또는 autoplay 속성의 결과로 paused 속성이 true에서 false로 변경될 때 스크립트 실행
onplaying	script	버퍼링을 위해 일시 중지 또는 중지된 후 다시 재생될 때 스크립트 실행
onprogress	script	브라우저가 미디어 데이터를 가져오는 중일 때 스크립트를 실행
onratechange	script	재생속도가 변명될 때마다 스크립트를 실행
onseeked	script	미디어의 재생 타임라인 변경이 끝났을 때 스크립트 실행
onseeking	script	미디어의 재생 타임라인을 변경 중일 때 스크립트 실행
onstalled	script	브라우저가 어떤 이유로든 미디어 데이터를 가져올 수 없을 때 스크립트 실행
ontimeupdate	script	재생위치가 변경 완료될 때 스크립트 실행 (예: 사용자가 미디어의 다른지점으로 빨리감기 하는 경우)
onvolumechange	script	미디어의 볼륨이 변경되었을 때 스크립트 실행
onwaiting	script	일시적인 재생할 데이터의 부족으로 인해 재생이 중지 되었을 때 스크립트 실행

# 4. Events

- HTML의 주요 공통 Event

## Misc Events

이벤트 속성명	값	설명
ontoggle	<i>script</i>	<details> 요소를 사용자가 열거나 닫을 때 스크립트 실행

## Window Events

이벤트 속성명	값	설명
onafterprint	<i>script</i>	문서 인쇄가 완료되었을 때 스크립트 실행
onbeforeprint	<i>script</i>	문서 인쇄되기 전 스크립트 실행
onbeforeunload	<i>script</i>	문서를 Unload(페이지 벗어남)하려고 할 때 스크립트 실행
onerror	<i>script</i>	오류가 발생했을 때 스크립트 실행
onload	<i>script</i>	페이지 로드가 완료되었을 때 스크립트 실행
onmessage	<i>script</i>	메시지가 트리거 될 때 스크립트 실행(다른 Window에서 메시지를 전송)
onpagehide	<i>script</i>	사용자가 페이지를 벗어날 때 스크립트 실행 (브라우저에서 뒤로가기 버튼 클릭 시)
onpageshow	<i>script</i>	사용자가 페이지로 이동할 때 스크립트 실행 (뒤로가기를 눌러 이전 페이지가 보여질 때)
onresize	<i>script</i>	브라우저 창의 크기를 조정할 때 스크립트 실행
onstorage	<i>script</i>	웹 저장소 영역이 업데이트 될 때 스크립트 실행
onunload	<i>script</i>	페이지가 언로드 되거나 브라우저 창이 닫힐 때 스크립트 실행



# CSS3

---

- 5. CSS란?
- 6. CSS Selector
- 7. CSS를 정의하는 3가지 방법
  - 8. Box Model
  - 9. Display Level
  - 10. Position
  - 11. Overflow
- 12. Transform, Transition

## 5. CSS란?

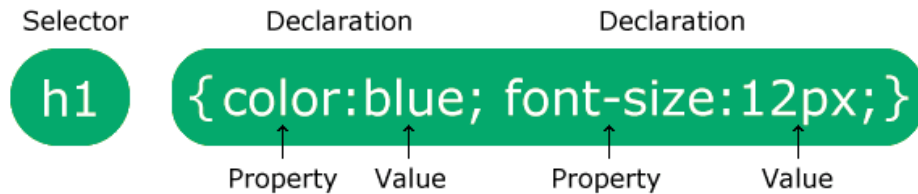
# 5. CSS란?

---

- CSS는 Cascading Style Sheets의 약자.
  - 계단식 스타일 문서.
    - 부모 태그에 CSS가 정의되면 자식 태그에도 동일하게 적용된다.
- CSS는 HTML 요소가 화면 또는 기타 미디어에 표시되는 방법을 작성.
- CSS는 많은 작업을 절약할 수 있다.
  - 한 번에 여러 웹 페이지의 레이아웃을 제어할 수 있다.
    - 웹 페이지에서 자주 사용되는 스타일을 한 번만 정의하고 재사용한다.
- 외부 스타일시트는 CSS파일 (.css파일)에 저장되어 사용한다.

# 5. CSS란?

## CSS Syntax



## CSS Example

```
body {  
  background-color: lightblue;  
}  
  
h1 {  
  color: white;  
  text-align: center;  
}  
  
p {  
  font-family: verdana;  
  font-size: 20px;  
}
```

## **6. CSS Selector**

# 6. CSS Selector

---

- Selector
  - 레이아웃을 정의할 때 CSS 가 HTML 태그를 찾아가는 방법
- Selector 종류
  - Simple selectors (태그명, ID, 클래스)
  - Combinator selectors (요소 간의 특정 관계에 따라 요소 선택)
  - Pseudo-class selectors (특정 상태에 따라 요소 선택)
  - Pseudo-elements selectors (요소의 일부 선택 및 스타일)
  - Attribute selectors (속성 또는 속성 을 기반으로 요소 선택)

## **6. CSS Selector**

- Simple selectors

# 6. CSS Selector - Simple selectors

---

- Element Selector

```
p {  
  text-align: center;  
  color: red;  
}
```

- ID Selector (#)

```
#para1 {  
  text-align: center;  
  color: red;  
}
```

- Class Selector (.)

```
.center {  
  text-align: center;  
  color: red;  
}
```

```
p.center {  
  text-align: center;  
  color: red;  
}
```



# 6. CSS Selector - Simple selectors

---

- Universal Selector (\*)

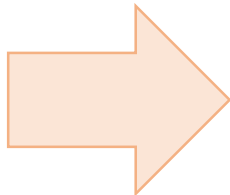
```
* {  
  text-align: center;  
  color: blue;  
}
```

- Grouping Selector (,)

```
h1 {  
  text-align: center;  
  color: red;  
}
```

```
h2 {  
  text-align: center;  
  color: red;  
}
```

```
p {  
  text-align: center;  
  color: red;  
}
```



```
h1, h2, p {  
  text-align: center;  
  color: red;  
}
```

## **6. CSS Selector**

- **Combinator selectors**

# 6. CSS Selector - Combinator selectors

- Descendant Selector (하위 셀렉터)
  - div 태그 안의 모든 p 태그

```
div p {  
  background-color: yellow;  
}
```

- Child Selector (>)
  - Div 태그 안의 첫 번째 자식 형제 p 태그

```
div > p {  
  background-color: yellow;  
}
```

- Adjacent Sibling Selector (+) (인접 형제 셀렉터)
  - div 태그 밑(형제) p태그

```
div + p {  
  background-color: yellow;  
}
```

# 6. CSS Selector - Combinator selectors

---

- General Sibling Selector (~) (일반 형제 셀렉터)
  - div 태그 뒤에 있는 형제 p 태그
  - 지정된 요소의 다음 형제인 모든 요소를 선택

```
div ~ p {  
  background-color: yellow;  
}
```

## **6. CSS Selector**

- Pseudo-class selectors

# 6. CSS Selector - Pseudo-class selectors

- Anchor Pseudo-classes
  - <a> 태그 스타일

```
/* unvisited link */
a:link {
  color: #FF0000;
}

/* visited link */
a:visited {
  color: #00FF00;
}

/* mouse over link */
a:hover {
  color: #FF00FF;
}

/* selected link */
a:active {
  color: #0000FF;
}
```

# 6. CSS Selector - Pseudo-class selectors

---

- Pseudo-classes and HTML Classes
  - Class가 “highlight”로 지정된 <a> 태그에 마우스를 올림

```
a.highlight:hover {  
  color: #ff0000;  
}
```

- Hover on <div>

```
div:hover {  
  background-color: blue;  
}
```

# 6. CSS Selector - Pseudo-class selectors

---

- Simple Tooltip Hover
  - Div 태그에 마우스를 올리면 p 태그가 나타남.

```
p {  
  display: none;  
  background-color: yellow;  
  padding: 20px;  
}  
  
div:hover p {  
  display: block;  
}
```



# 6. CSS Selector - Pseudo-class selectors

---

- CSS - The :first-child Pseudo-class
- Match the first <p> element

```
p:first-child {  
  color: blue;  
}
```

- Match the first <i> element in all <p> elements

```
p i:first-child {  
  color: blue;  
}
```

- Match all <i> elements in all first child <p> elements

```
p:first-child i {  
  color: blue;  
}
```

# 6. CSS Selector - Pseudo-class selectors

- All CSS Pseudo Classes ( 1 / 2)

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links

# 6. CSS Selector - Pseudo-class selectors

- All CSS Pseudo Classes ( 2 / 2 )

Selector	Example	Example description
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

## **6. CSS Selector**

- Pseudo-elements selectors

# 6. CSS Selector

## - Pseudo-elements selectors

---

- `::first-line` Pseudo-element
  - Block 엘리먼트에서만 사용할 수 있음.
    - `div`, `p` 등

```
p::first-line {  
  color: #ff0000;  
  font-variant: small-caps;  
}
```

- `::first-letter` Pseudo-element
  - Block 엘리먼트에서만 사용할 수 있음.
    - `div`, `p` 등

```
p::first-letter {  
  color: #ff0000;  
  font-size: xx-large;  
}
```

# 6. CSS Selector

## - Pseudo-elements selectors

---

- Pseudo-elements and HTML Classes
  - Class가 “intro” 인 <p> 태그 내용 중 첫 번째 글자

```
p.intro::first-letter {  
  color: #ff0000;  
  font-size: 200%;  
}
```

- Multiple Pseudo-elements
  - 첫 번째 줄을 푸른색으로, 첫 번째 줄의 첫 번째 글자를 붉은 색으로 변경

```
p::first-letter {  
  color: #ff0000;  
  font-size: xx-large;  
}  
  
p::first-line {  
  color: #0000ff;  
  font-variant: small-caps;  
}
```

# 6. CSS Selector

## - Pseudo-elements selectors

---

- ::after Pseudo-element
  - H1 엘리먼트 뒤에 이미지를 삽입.

```
h1::after {  
  content: url(smiley.gif);  
}
```

- ::marker Pseudo-element
  - li 태그의 마커(순서, 도형 등)의 스타일을 변경

```
::marker {  
  color: red;  
  font-size: 23px;  
}
```

# 6. CSS Selector

## - Pseudo-elements selectors

---

- ::selection Pseudo-element
  - 드래그 한 글자의 배경색과 글자색을 변경

```
::selection {  
  color: red;  
  background: yellow;  
}
```



## **6. CSS Selector**

- **Attribute selectors**

# 6. CSS Selector - Attribute selectors

- [attribute] Selector
  - Target 속성이 부여된 <a> 태그의 스타일을 변경

```
a[target] {  
  background-color: yellow;  
}
```

- [attribute="value"] Selector
  - Target 속성의 값이 "\_blank" 인 <a> 태그의 스타일을 변경

```
a[target="_blank"] {  
  background-color: yellow;  
}
```

- [attribute~="value"] Selector
  - title 속성의 값이 "flower" 로 끝나는 모든 태그의 스타일을 변경

```
[title~="flower"] {  
  border: 5px solid yellow;  
}
```

# 6. CSS Selector - Attribute selectors

- [attribute]="value" Selector
  - Class 속성의 값이 "top" 이거나 "top-"로 시작하는 모든 태그들의 스타일을 변경

```
[class]="top" {  
  background: yellow;  
}
```

- [attribute^="value"] Selector
  - Class 속성의 값이 "top"으로 시작하는 모든 태그들의 스타일을 변경

```
[class^="top"] {  
  background: yellow;  
}
```

- [attribute\$="value"] Selector
  - Class 속성의 값이 "test"으로 끝나는 모든 태그들의 스타일을 변경

```
[class$="test"] {  
  background: yellow;  
}
```

# 6. CSS Selector - Attribute selectors

- [attribute\*="value"] Selector
  - Class 속성의 값에 "te" 가 포함된 모든 태그들의 스타일을 변경

```
[class*="te"] {  
  background: yellow;  
}
```

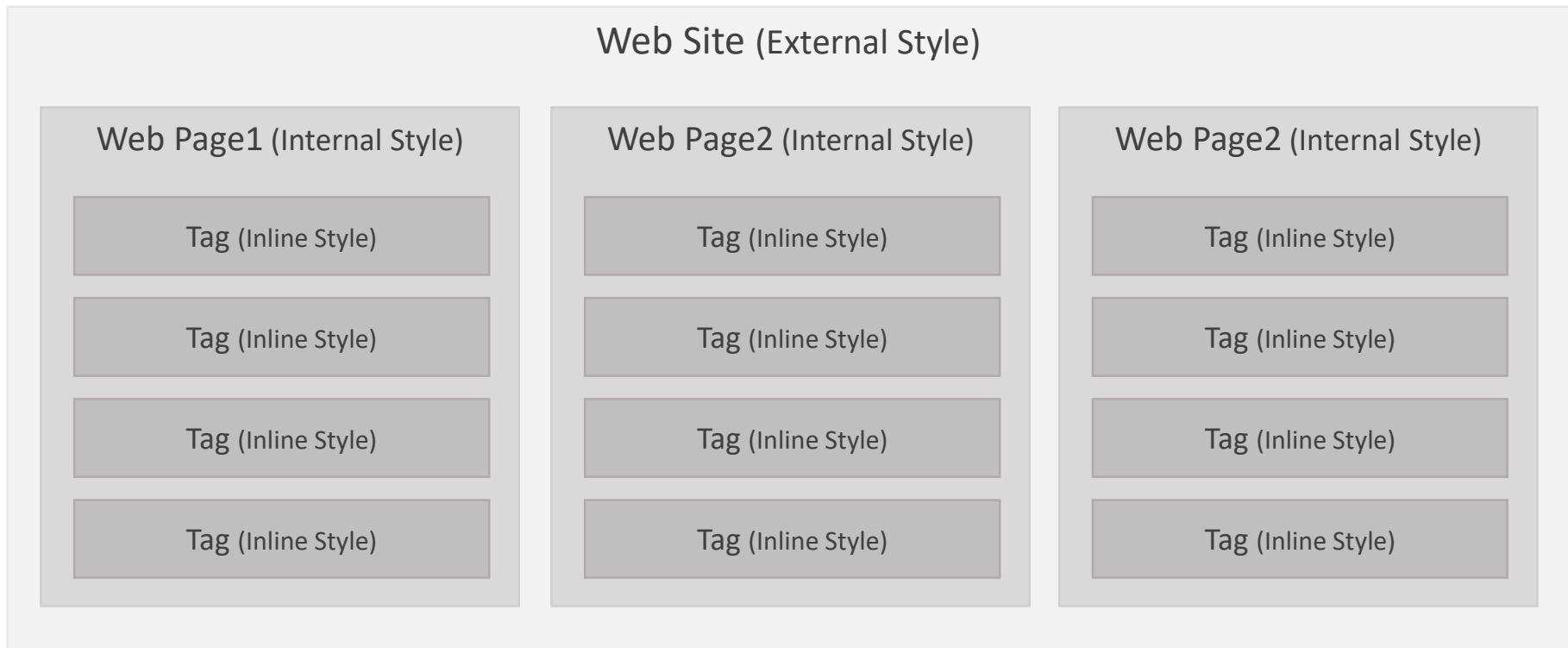
- Styling Forms

```
input[type="text"] {  
  width: 150px;  
  display: block;  
  margin-bottom: 10px;  
  background-color: yellow;  
}  
  
input[type="button"] {  
  width: 120px;  
  margin-left: 35px;  
  display: block;  
}
```

## **7. CSS를 정의하는 3가지 방법**

# 7. CSS를 정의하는 3가지 방법

- CSS는 적용 범위에 따라 3가지 방법으로 작성할 수 있다.
  - 웹 사이트의 전체 레이아웃을 적용할 수 있는 External Style
  - 한 페이지의 레이아웃만 적용할 수 있는 Internal Style
  - 한 태그의 레이아웃만 적용할 수 있는 Inline Style



## 7. CSS를 정의하는 3가지 방법

- External Style

# 7. CSS를 정의하는 3가지 방법

## - External Style

- 별도의 CSS파일을 만들어 필요한 웹 페이지에 로드하는 방법.
- 보편적으로 사용하는 방법 중 하나
  - 관련된 태그에 대한 스타일을 하나로 모아 관리. ➔ 스타일 수정이나 추가가 용이.
    - 예> 테이블 스타일, 단락 스타일, 헤더, 푸터, 메뉴 스타일 등.
  - CSS 파일만 작성하므로 변경 이력 추적에도 용이.
    - CSS 파일을 버전별로 만들어 사용할 수도 있다.

"mystyle.css"

```
body {  
  background-color: lightblue;  
}  
  
h1 {  
  color: navy;  
  margin-left: 20px;  
}
```

```
<!DOCTYPE html>  
<html>  
  <head>  
    <link rel="stylesheet" href="mystyle.css">  
  </head>  
  <body>  
  
    <h1>This is a heading</h1>  
    <p>This is a paragraph.</p>  
  
  </body>  
</html>
```



## 7. CSS를 정의하는 3가지 방법

- Internal Style

# 7. CSS를 정의하는 3가지 방법

## - Internal Style

- 웹 페이지의 <style> 태그 안에 스타일을 정의하는 방법
  - External Style의 내용 중 일부를 추가/수정 하고자 할 때 사용

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

## 7. CSS를 정의하는 3가지 방법

- Inline Style

# 7. CSS를 정의하는 3가지 방법

## - Internal Style

---

- 태그에 직접 스타일을 추가하는 방법
  - 스타일 추적이 어렵기 때문에 권장하지 않는 방법.

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

## 7. CSS를 정의하는 3가지 방법

- 스타일별 우선순위

# 7. CSS를 정의하는 3가지 방법

## - 스타일별 우선순위

- Cascading Order (우선순위가 존재하지만, HTML 코드의 순서에 따라 달라진다.)
  - 1. Inline Style
  - 2. Internal Style
  - 3. External Style

"mystyle.css"

```
body {  
  background-color: lightblue;  
}
```

```
h1 {  
  color: navy;  
  margin-left: 20px;  
}
```

```
<head>  
<link rel="stylesheet" type="text/css" href="mystyle.css">  
<style>  
  h1 {  
    color: orange;  
  }  
</style>  
</head>
```

External Style + Internal Style  
H1 color is orange

```
<head>  
<style>  
  h1 {  
    color: orange;  
  }  
</style>  
<link rel="stylesheet" type="text/css" href="mystyle.css">  
</head>
```

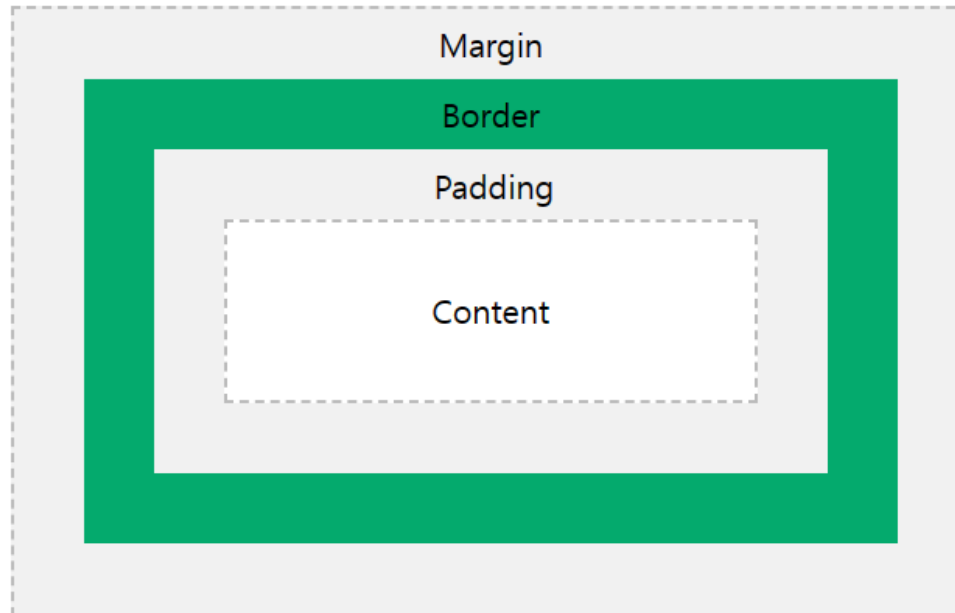
Internal Style + External Style  
H1 color is navy

## **8. CSS Box Model**

# 8. CSS Box Model

---

- 기본적으로 HTML 요소를 감싸는 상자.
- 여백, 테두리, 패딩 및 실제 콘텐츠로 구성된다.

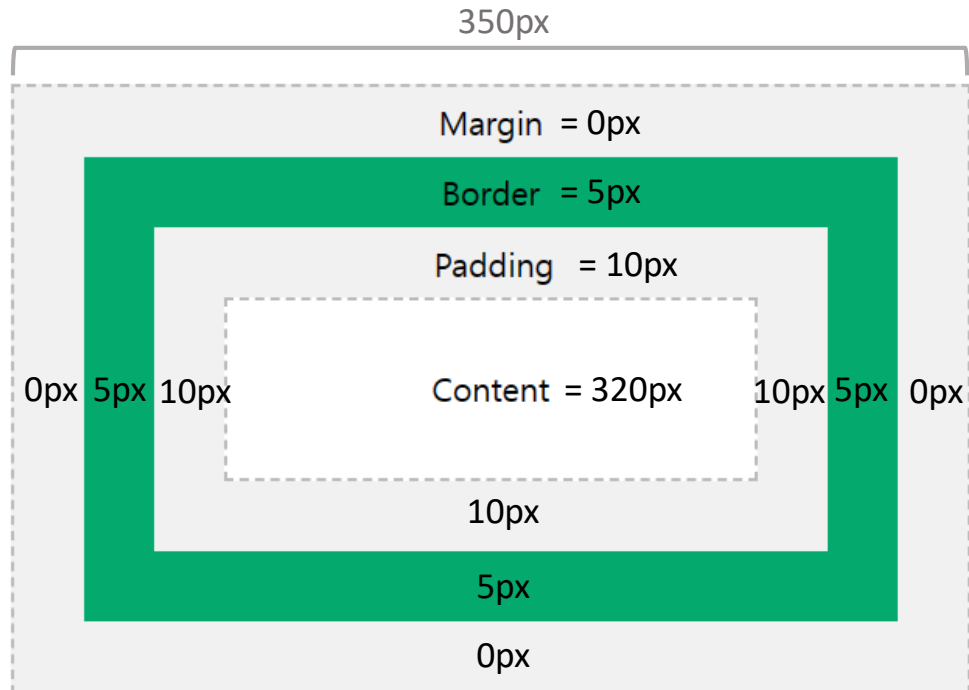




# 8. CSS Box Model

- Box Model Styling

```
div {  
  width: 320px;  
  padding: 10px;  
  border: 5px solid gray;  
  margin: 0;  
}
```

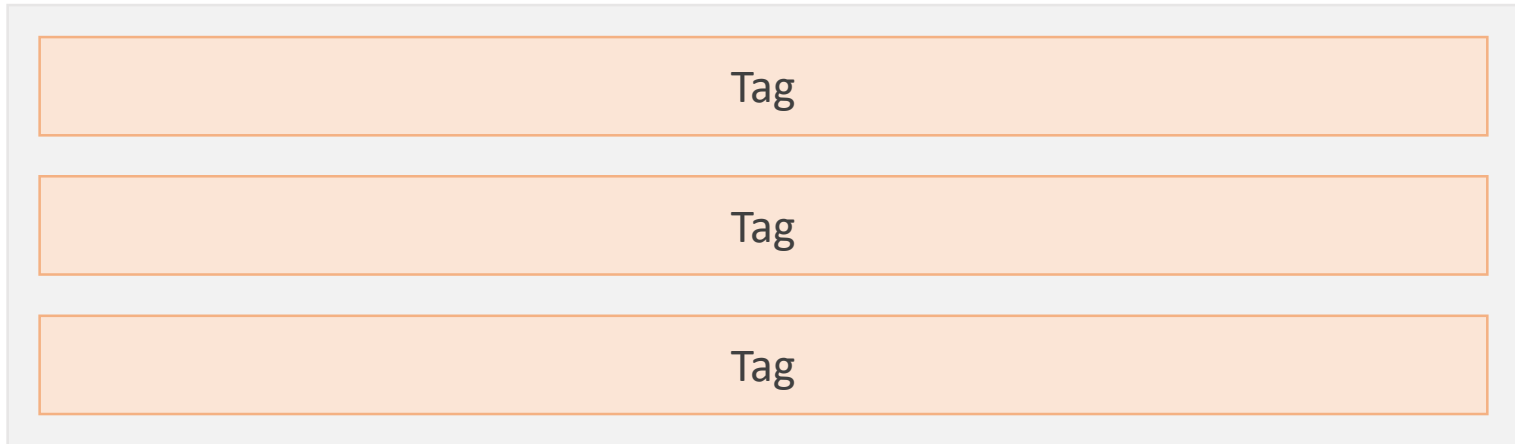


## 9. Display Level

# 9. Display Level

---

- Block Display Tag
  - 엘리먼트가 새 줄에서 시작해서 전체 너비를 차지함.



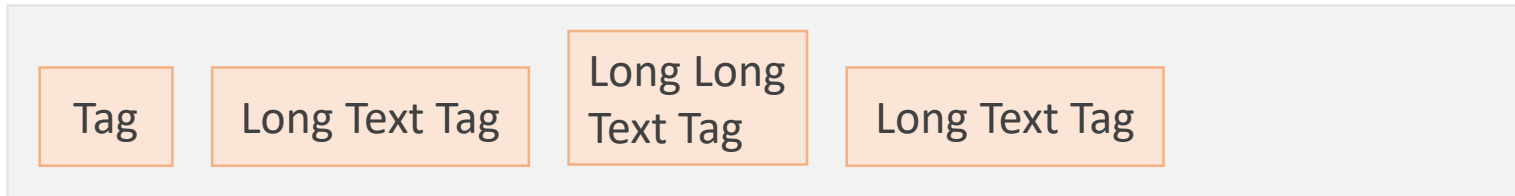
- Inline Display Tag
  - 엘리먼트가 콘텐츠 영역만큼만 차지함
  - 단, width와 height는 적용할 수 없음.
  - 콘텐츠가 <br/> 등으로 밀려나면 새 줄의 처음부터 시작함.



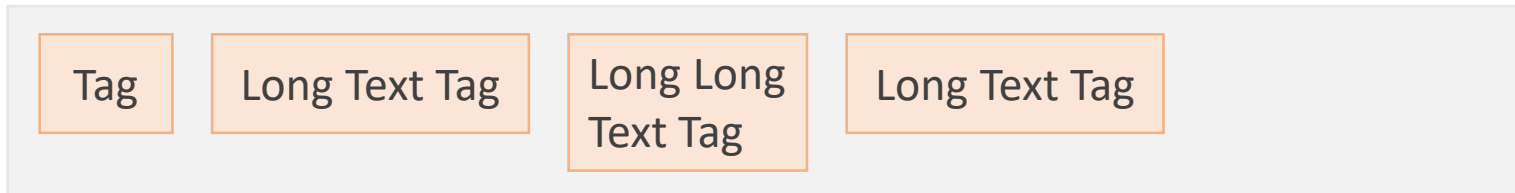
# 9. Display Level

---

- Inline-block Display Tag
  - 엘리먼트가 콘텐츠의 영역만큼 차지함.
  - width와 height 적용이 가능함.
  - 콘텐츠가 <br/> 등으로 밀려나면 블록 영역에서 개행됨.



- display: inline-block;  
vertical-align: top;



# 9. Display Level

- Display의 값 목록

## Display General Value

Display value	Description
inline	요소를 인라인 요소(예: <span>)로 표시합니다. 높이 및 너비 속성은 영향을 주지 않습니다
block	요소를 블록 요소(예: <p>)로 표시합니다. 새 줄에서 시작하여 전체 너비를 차지합니다
inline-block	요소를 인라인 수준 블록 컨테이너로 표시합니다. 요소 자체는 인라인 요소로 형식이 지정되지만 높이 및 너비 값을 적용할 수 있습니다
inline-table	요소는 인라인 수준 테이블로 표시됩니다
table	요소가 <table>요소처럼 동작하도록 합니다.
table-header-group	요소가 <thead>요소처럼 동작하도록 합니다.
table-row-group	요소가 <tbody>요소처럼 동작하도록 합니다.
table-cell	요소가 <td>요소처럼 동작하도록 합니다.
table-row	요소가 <tr>요소처럼 동작하도록 합니다.
none	요소가 완전히 제거됩니다

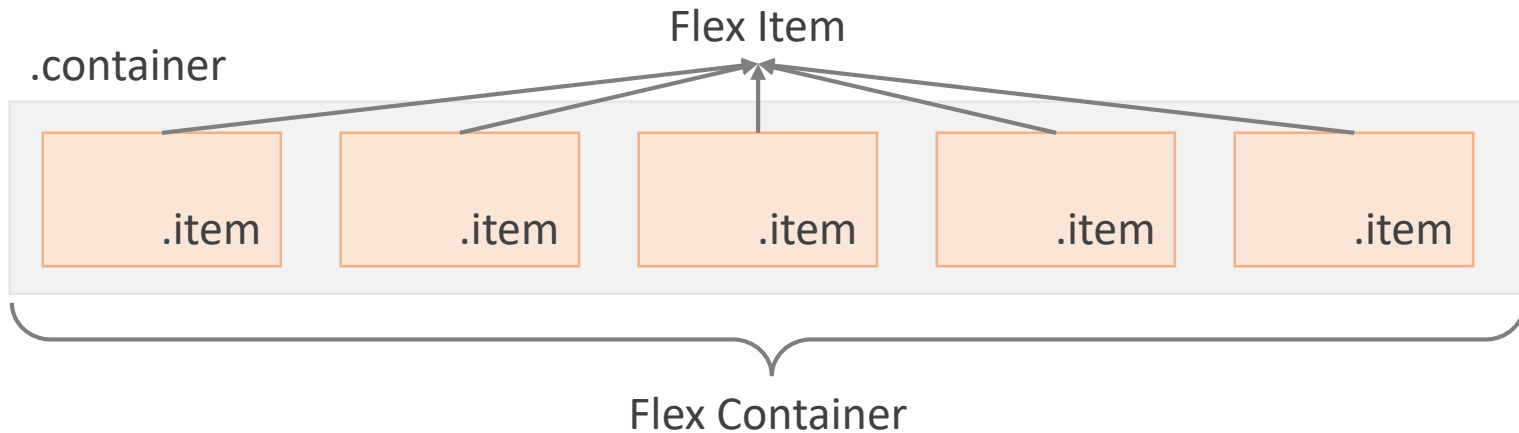
## Display Container System Value

Display value	Description
flex	요소를 블록 수준 플렉스 컨테이너로 표시합니다
grid	요소를 블록 수준 그리드 컨테이너로 표시합니다
inline-flex	요소를 인라인 수준 플렉스 컨테이너로 표시합니다
inline-grid	요소를 인라인 수준 그리드 컨테이너로 표시합니다

## **9. Display Level – Flex Container**

# 9. Display Level – Flex Container

- Display container system
  - Flex (Flexible Box, FlexBox)
  - 레이아웃 배치 전용 기능.
  - Flex는 Container와 Item에 대한 속성을 부여함으로써 레이아웃을 정의한다.



# 9. Display Level - Flex Container

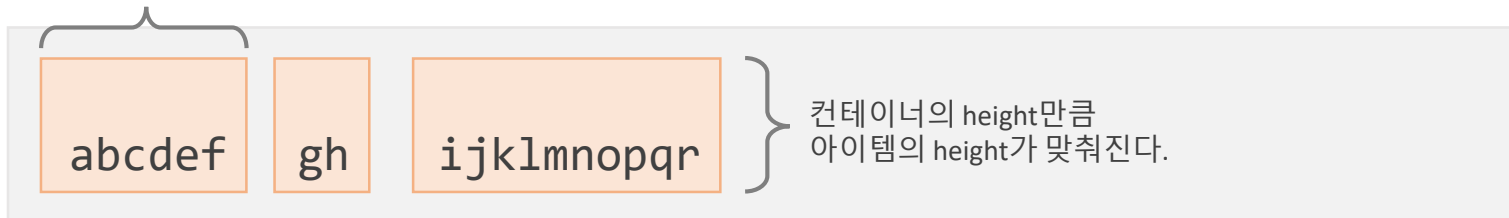
- Flex container system

- Style

```
.container {  
  display: flex;  
  /* display: inline-flex; */  
}
```

```
<div class="container">  
  <div class="item">abcdef</div>  
  <div class="item">gh</div>  
  <div class="item">ijklmnopqr</div>  
</div>
```

내용물의 길이만큼만  
너비가 지정된다.



교차 축  
혹은  
수직 축  
(Cross Axis)

메인 축 (Main Axis)



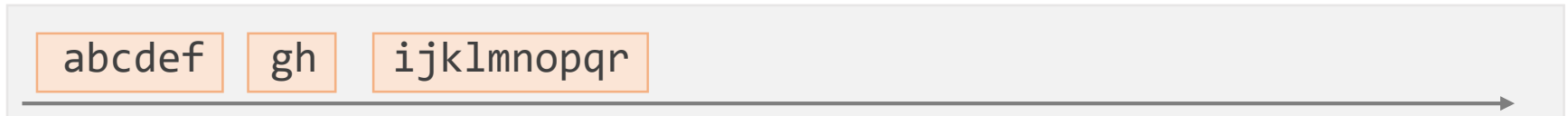
# 9. Display Level - Flex Container

- 배치 방향 설정

- flex-direction

```
.container {  
  display: flex;  
  flex-direction: row;  
  /* flex-direction: column; */  
  /* flex-direction: row-reverse; */  
  /* flex-direction: column-reverse; */  
}
```

- row



- row-reverse



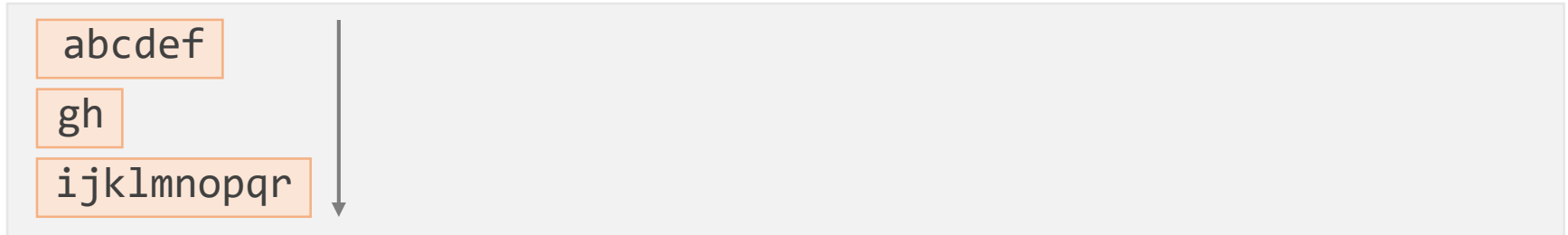
# 9. Display Level - Flex Container

- 배치 방향 설정

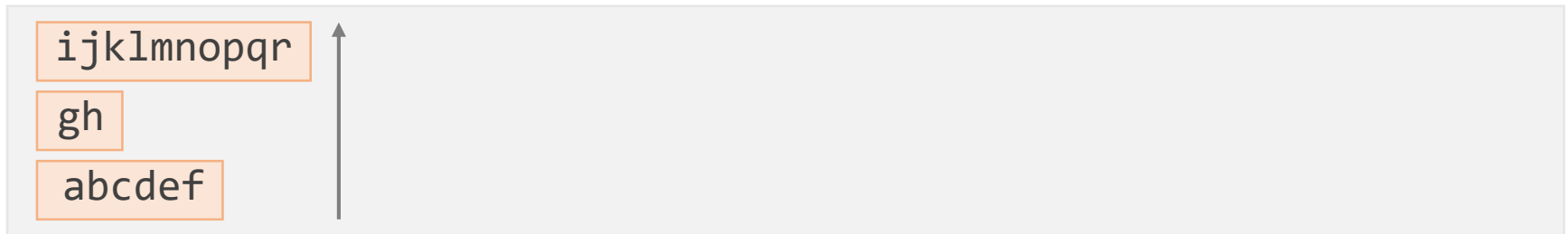
- flex-direction

```
.container {  
  display: flex;  
  flex-direction: row;  
  /* flex-direction: column; */  
  /* flex-direction: row-reverse; */  
  /* flex-direction: column-reverse; */  
}
```

- column



- column-reverse



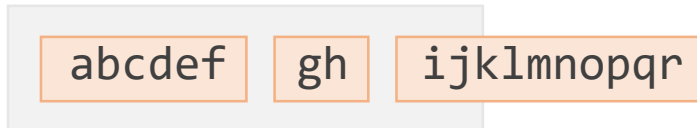
# 9. Display Level - Flex Container

- 줄 넘김 처리 설정

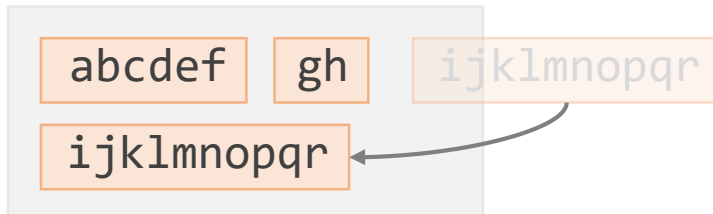
- flex-wrap

```
.container {  
  display: flex;  
  flex-wrap: nowrap;  
  /* flex-wrap: wrap; */  
  /* flex-wrap: wrap-reverse; */  
}
```

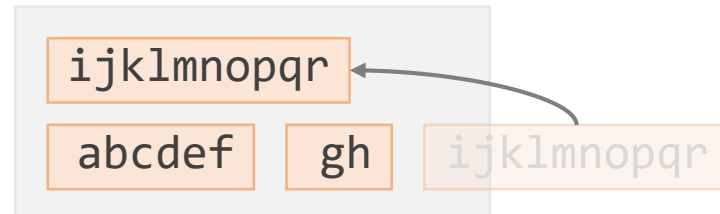
- nowrap



- wrap



- wrap-reverse



# 9. Display Level - Flex Container

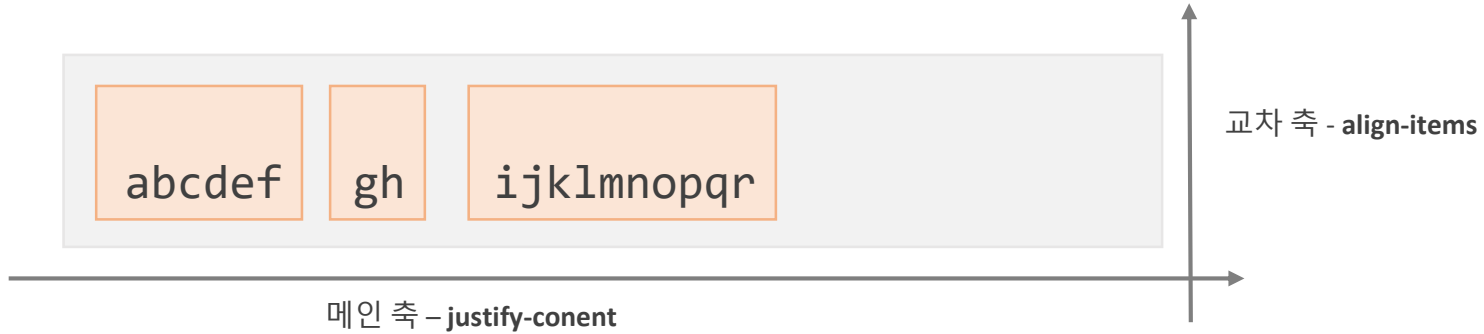
---

- 줄 넘김 처리 설정 (flex-direction + flex-wrap)
  - flex-wrap (flex-direction flex-wrap 순으로 띄어쓰기로 구분해 작성)

```
.container {  
  display: flex;  
  flex-wrap: nowrap;  
  /* flex-wrap: wrap; */  
  /* flex-wrap: wrap-reverse; */  
}
```

# 9. Display Level - Flex Container

- 정렬 설정
  - Flex의 정렬 방법



# 9. Display Level - Flex Container

- 메인 축 정렬

- justify-content

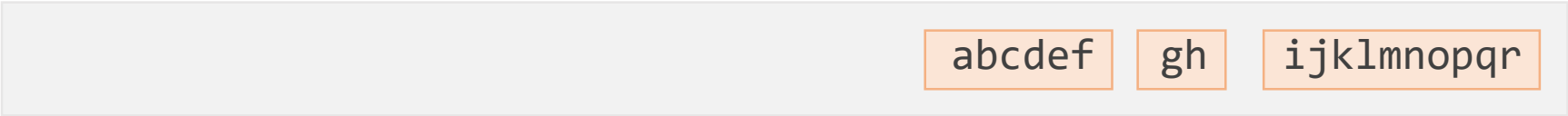
```
.container {  
  display: flex;  
  justify-content: flex-start;  
  /* justify-content: flex-end; */  
  /* justify-content: center; */  
  /* justify-content: space-between; */  
  /* justify-content: space-around; */  
  /* justify-content: space-evenly; */  
}
```

- flex-start – 아이টে를 시작점으로 정렬



abcdef gh ijklmnopqr

- flex-end – 아이টে를 끝점으로 정렬



abcdef gh ijklmnopqr

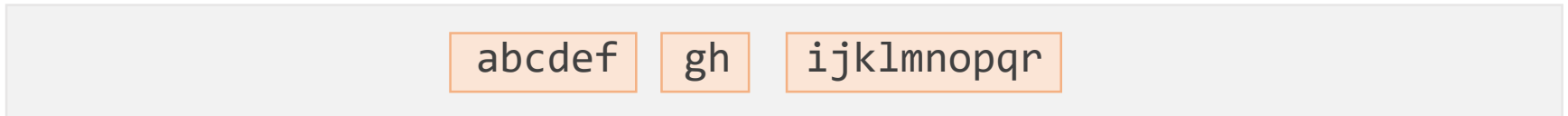
# 9. Display Level - Flex Container

- 메인 축 정렬

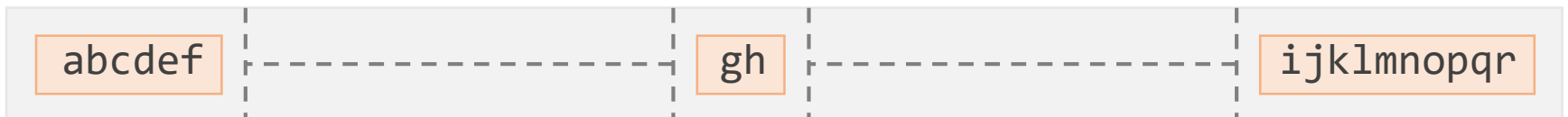
- justify-content

```
.container {  
  display: flex;  
  justify-content: flex-start;  
  /* justify-content: flex-end; */  
  /* justify-content: center; */  
  /* justify-content: space-between; */  
  /* justify-content: space-around; */  
  /* justify-content: space-evenly; */  
}
```

- center – 아이টে를 가운데로 정렬



- space-between – 아이টে를 사이(between)에 균일한 간격을 만들어 줌



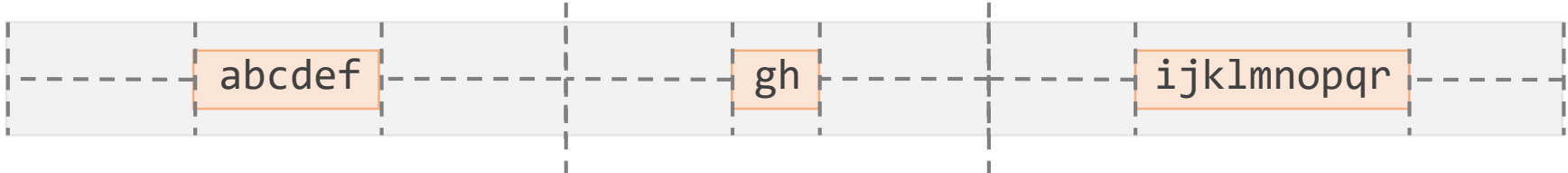
# 9. Display Level - Flex Container

- 메인 축 정렬

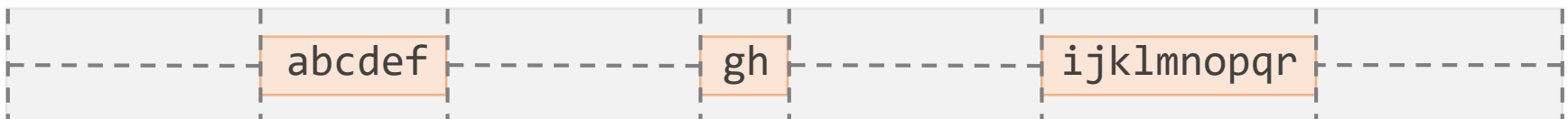
- justify-content

```
.container {  
  display: flex;  
  justify-content: flex-start;  
  /* justify-content: flex-end; */  
  /* justify-content: center; */  
  /* justify-content: space-between; */  
  /* justify-content: space-around; */  
  /* justify-content: space-evenly; */  
}
```

- space-around – 아이탬들 둘레(around)에 균일한 간격을 만들어 줌



- space-evenly – 아이탬들 사이와 양끝에 균일한 간격을 만들어 줌





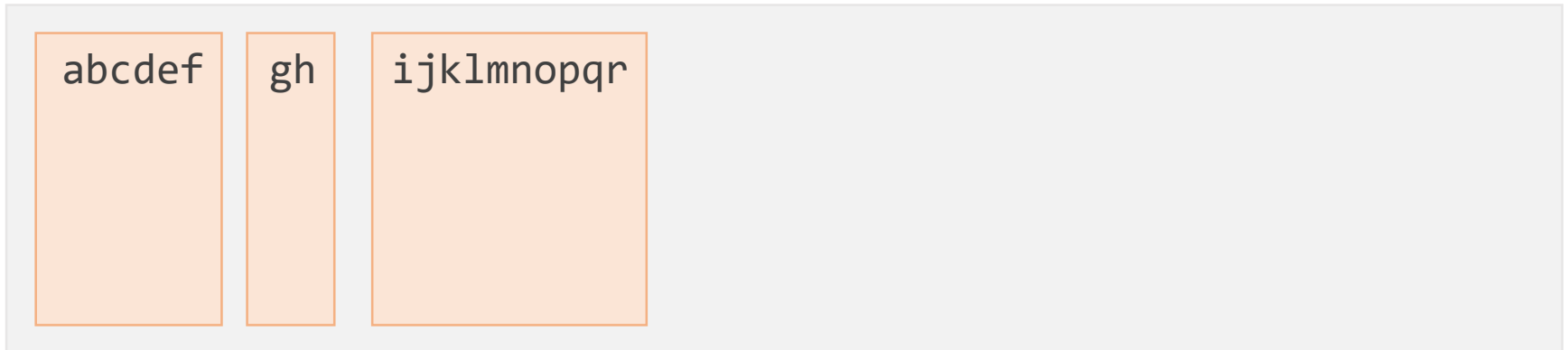
# 9. Display Level - Flex Container

- 수직 축 정렬

- align-items

```
.container {  
  display: flex;  
  height: 200px;  
  align-items: stretch;  
  /* align-items: flex-start; */  
  /* align-items: flex-end; */  
  /* align-items: center; */  
  /* align-items: baseline; */  
}
```

- stretch – 아이템을 수직 축 방향으로 쭉 늘어트림



# 9. Display Level - Flex Container

- 수직 축 정렬

- align-items

```
.container {  
  display: flex;  
  height: 200px;  
  align-items: stretch;  
  /* align-items: flex-start; */  
  /* align-items: flex-end; */  
  /* align-items: center; */  
  /* align-items: baseline; */  
}
```

- flex-start – 아이টে를 시작점으로 정렬

abcdef

gh

ijklmnopqr

# 9. Display Level - Flex Container

- 수직 축 정렬

- align-items

```
.container {  
  display: flex;  
  height: 200px;  
  align-items: stretch;  
  /* align-items: flex-start; */  
  /* align-items: flex-end; */  
  /* align-items: center; */  
  /* align-items: baseline; */  
}
```

- flex-end – 아이টে를 끝점으로 정렬

abcdef

gh

ijklmnopqr

# 9. Display Level - Flex Container

- 수직 축 정렬

- align-items

```
.container {  
  display: flex;  
  height: 200px;  
  align-items: stretch;  
  /* align-items: flex-start; */  
  /* align-items: flex-end; */  
  /* align-items: center; */  
  /* align-items: baseline; */  
}
```

- center – 아이টে을 가운데로 정렬

abcdef

gh

ijklmnopqr

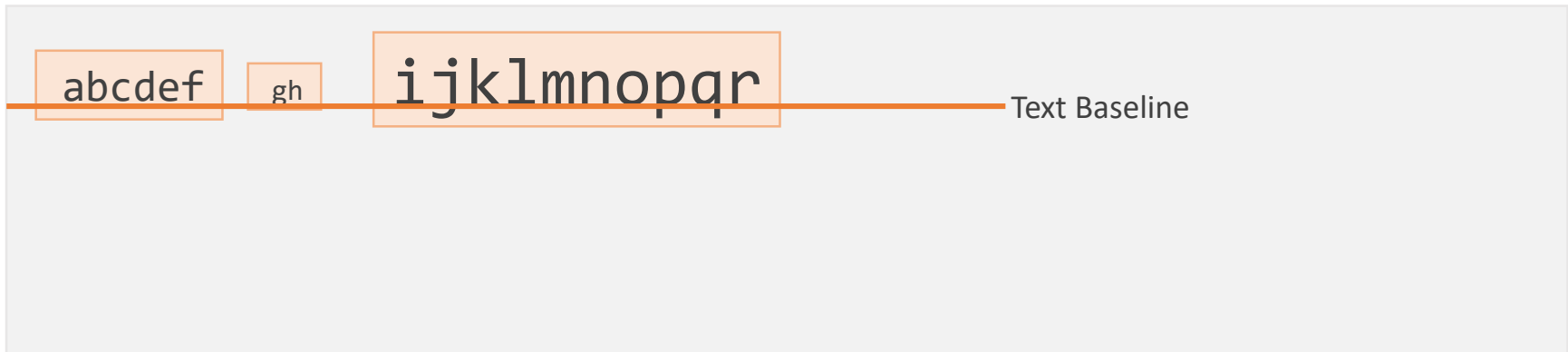
# 9. Display Level - Flex Container

- 수직 축 정렬

- align-items

```
.container {  
  display: flex;  
  height: 200px;  
  align-items: stretch;  
  /* align-items: flex-start; */  
  /* align-items: flex-end; */  
  /* align-items: center; */  
  /* align-items: baseline; */  
}
```

- baseline – 아이тем들을 텍스트 베이스라인 기준으로 정렬



# 9. Display Level - Flex Container

- 메인, 수직 축 정렬

- justify-content, align-items

```
.container {  
  display: flex;  
  height: 200px;  
  justify-content: center;  
  align-items: center;  
}
```

- center – 아이تم들을 정 가운데로 정렬

abcdef

gh

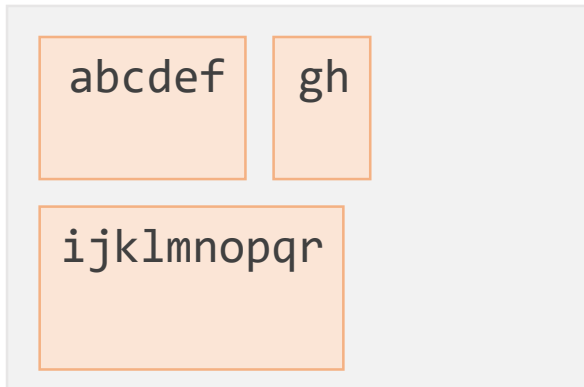
ijklmnopqr

# 9. Display Level - Flex Container

- 여러 행 정렬
  - align-content
    - flex-wrap: wrap;이 설정된 상태에서,  
아이템들의 행이 2줄 이상 되었을 때의 수직축 방향 정렬을 결정하는 속성

```
.container {  
  display: flex;  
  width: 700px;  
  height: 500px;  
  flex-wrap: wrap;  
  align-content: stretch;  
}
```

- stretch – 아이템을 수직 축 방향으로 짝 늘어트림

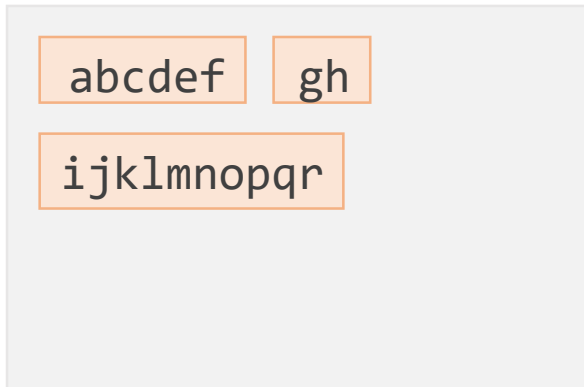


# 9. Display Level - Flex Container

- 여러 행 정렬
  - align-content
    - flex-wrap: wrap;이 설정된 상태에서,  
아이템들의 행이 2줄 이상 되었을 때의 수직축 방향 정렬을 결정하는 속성

```
.container {  
  display: flex;  
  width: 700px;  
  height: 500px;  
  flex-wrap: wrap;  
  align-content: flex-start;  
}
```

- flex-start – 아이템을 시작점으로 정렬



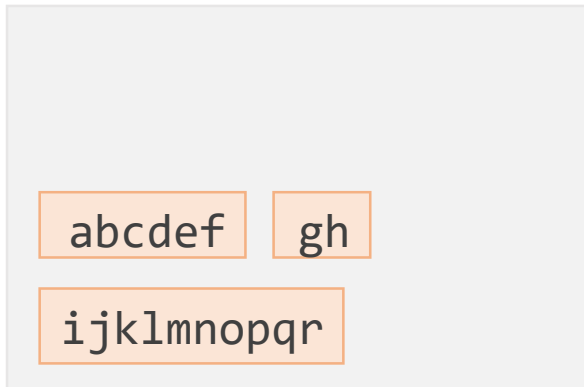


# 9. Display Level - Flex Container

- 여러 행 정렬
  - align-content
    - flex-wrap: wrap;이 설정된 상태에서,  
아이템들의 행이 2줄 이상 되었을 때의 수직축 방향 정렬을 결정하는 속성

```
.container {  
  display: flex;  
  width: 700px;  
  height: 500px;  
  flex-wrap: wrap;  
  align-content: flex-end;  
}
```

- flex-end – 아이템들을 끝으로 정렬

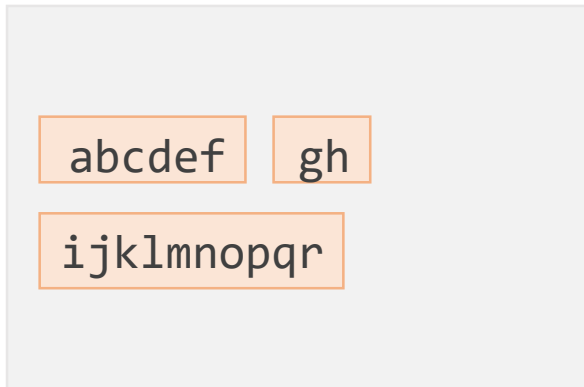


# 9. Display Level - Flex Container

- 여러 행 정렬
  - align-content
    - flex-wrap: wrap;이 설정된 상태에서,  
아이템들의 행이 2줄 이상 되었을 때의 수직축 방향 정렬을 결정하는 속성

```
.container {  
  display: flex;  
  width: 700px;  
  height: 500px;  
  flex-wrap: wrap;  
  align-content: center;  
}
```

- center – 아이템들을 가운데로 정렬

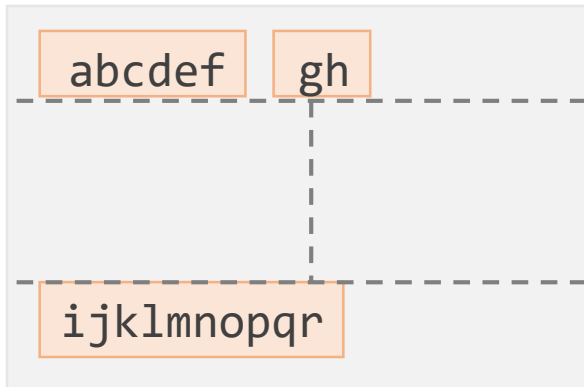


# 9. Display Level - Flex Container

- 여러 행 정렬
  - align-content
    - flex-wrap: wrap;이 설정된 상태에서,  
아이템들의 행이 2줄 이상 되었을 때의 수직축 방향 정렬을 결정하는 속성

```
.container {  
  display: flex;  
  width: 700px;  
  height: 500px;  
  flex-wrap: wrap;  
  align-content: space-between;  
}
```

- space-between – 아이템들의 사이(between)에 균일한 간격을 만들어 줌

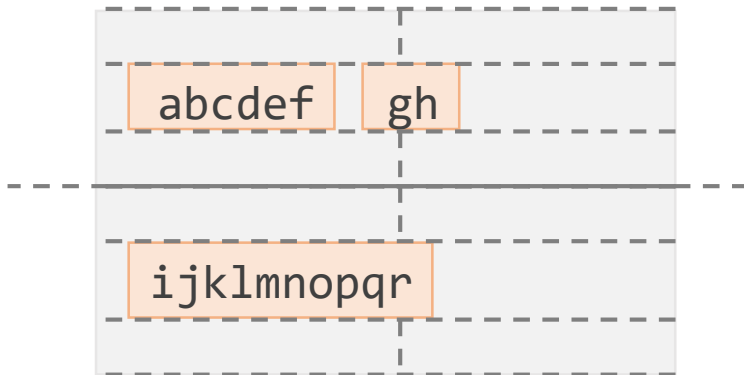


# 9. Display Level - Flex Container

- 여러 행 정렬
  - align-content
    - flex-wrap: wrap;이 설정된 상태에서,  
아이템들의 행이 2줄 이상 되었을 때의 수직축 방향 정렬을 결정하는 속성

```
.container {  
  display: flex;  
  width: 700px;  
  height: 500px;  
  flex-wrap: wrap;  
  align-content: space-around;  
}
```

- space-around – 아이템들의 둘레(around)에 균일한 간격을 만들어 줌

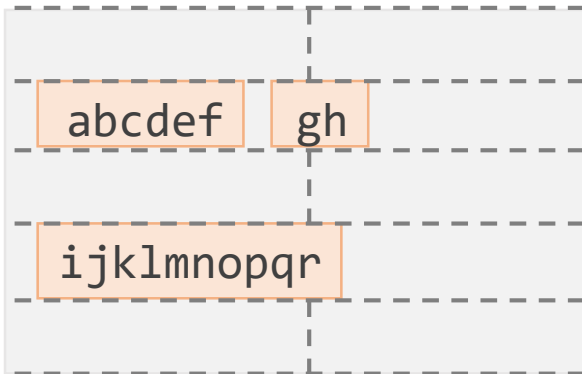


# 9. Display Level - Flex Container

- 여러 행 정렬
  - align-content
    - flex-wrap: wrap;이 설정된 상태에서,  
아이템들의 행이 2줄 이상 되었을 때의 수직축 방향 정렬을 결정하는 속성

```
.container {  
  display: flex;  
  width: 700px;  
  height: 500px;  
  flex-wrap: wrap;  
  align-content: space-evenly;  
}
```

- space-evenly – 아이템들의 사이와 양 끝에 균일한 간격을 만들어 줌



## **9. Display Level – Flex Item**

# 9. Display Level - Flex Item

- Flex Item의 기본 크기 설정
  - flex-basis
    - flex-direction이 row일 때는 너비
    - flex-direction이 column일 때는 높이

```
.item {  
  flex-basis: auto; /* 기본값 */  
}
```

- auto, content - 컨텐츠 크기에 따라 자동 조절

abcdef

gh

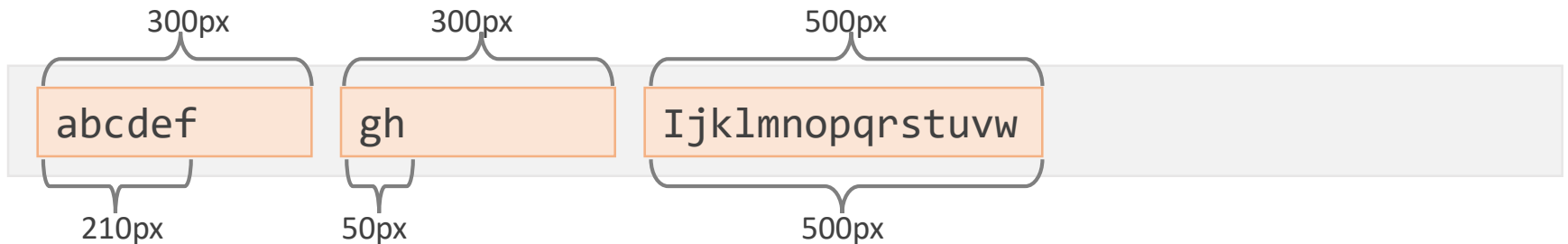
ijklmnopqr

# 9. Display Level - Flex Item

- Flex Item의 기본 크기 설정
  - flex-basis
    - flex-direction이 row일 때는 너비
    - flex-direction이 column일 때는 높이

```
.item {  
  flex-basis: 300px;  
}
```

- 300px – item 의 기본 크기를 300px로 맞춤.  
300px이 되지 않는 아이템의 크기는 300px로 맞춰지고,  
초과하는 item은 원래 크기 유지됨.



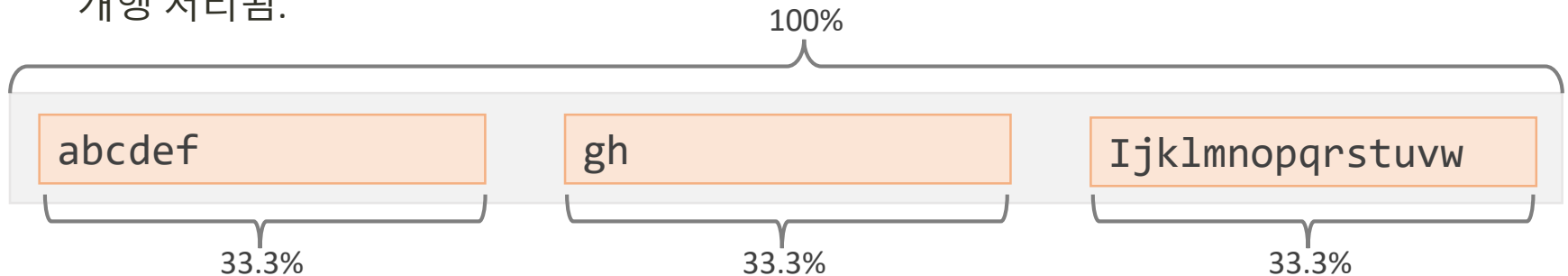


# 9. Display Level - Flex Item

- Flex Item의 기본 크기 설정
  - flex-basis
    - flex-direction이 row일 때는 너비
    - flex-direction이 column일 때는 높이

```
.item {  
  flex-basis: 50%;  
}
```

- 50% – item의 기본 크기를 윈도우 사이즈의 50%로 맞춤.  
Item이 나열된 크기가 100% 넘어갈 경우 flex container의 flex-wrap 설정 여부에 따라  
개행 처리됨.



# 9. Display Level - Flex Item

- Flex Item의 기본 크기 설정

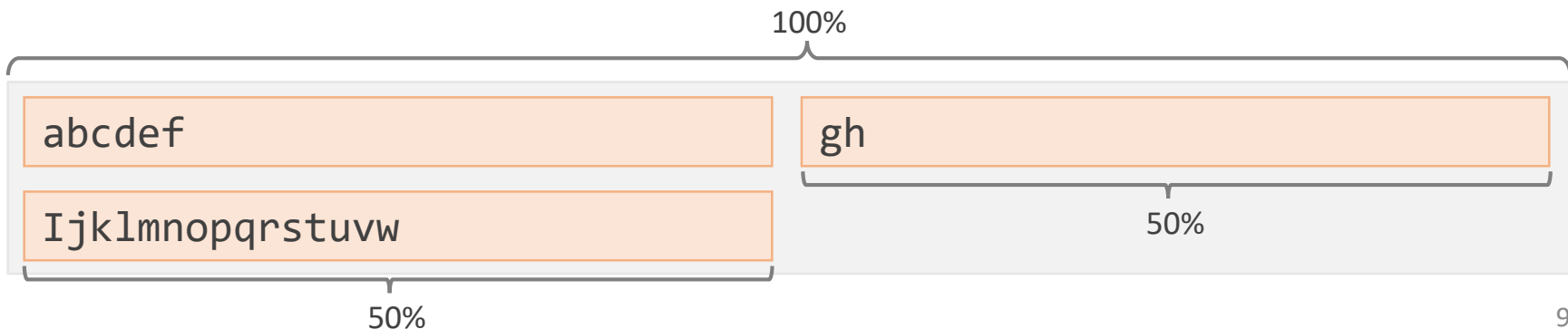
- flex-basis

- flex-direction이 row일 때는 너비
    - flex-direction이 column일 때는 높이

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

```
.item {  
  flex-basis: 50%;  
}
```

- 50% – item의 기본 크기를 윈도우 사이즈의 50%로 맞춤.  
Item이 나열된 크기가 100% 넘어갈 경우 flex container의 flex-wrap 설정 여부에 따라  
개행 처리됨.

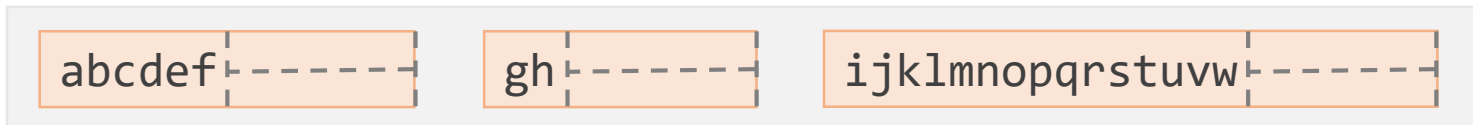


# 9. Display Level - Flex Item

- Flex Item을 유연하게 늘리기
  - flex-grow (기본 값: 0)
    - flex-basis의 값보다 커질 수 있는지 결정하는 속성
    - flex-grow의 값이 0보다 클 경우, flex container의 빈 공간을 모두 메우게 됨.

```
.item {  
    flex-grow: 1;  
}
```

- flex-grow의 값이 1인 경우, flex container의 남은 여백을 균일하게 채운다.



# 9. Display Level - Flex Item

- Flex Item을 유연하게 늘리기
  - flex-grow (기본 값: 0)
    - flex-basis의 값보다 커질 수 있는지 결정하는 속성
    - flex-grow의 값이 0보다 클 경우, flex container의 빈 공간을 모두 메우게 됨.

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

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

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

- flex-grow의 값이 모두 다를 경우 grow의 비율만큼 여백을 채운다.



# 9. Display Level - Flex Item

---

- Flex Item을 유연하게 줄이기
  - flex-shrink (기본 값: 1)
    - flex-grow와 쌍을 이루는 속성.
    - flex-basis보다 작아질 수 있는지 결정하는 속성
    - flex-shrink의 값이 0보다 클 경우 flex-basis보다 작아질 수 있다.

```
.container {  
  display: flex;  
  width: 250px  
}  
  
.item:nth-child(1) {  
  flex-shrink: 0;  
  width: 100px;  
}  
  
.item:nth-child(2) {  
  flex-grow: 1;  
}  
  
.item:nth-child(3) {  
  flex-grow: 1;  
}
```

# 9. Display Level - Flex Item

- Flex Item을 유연하게 줄이기

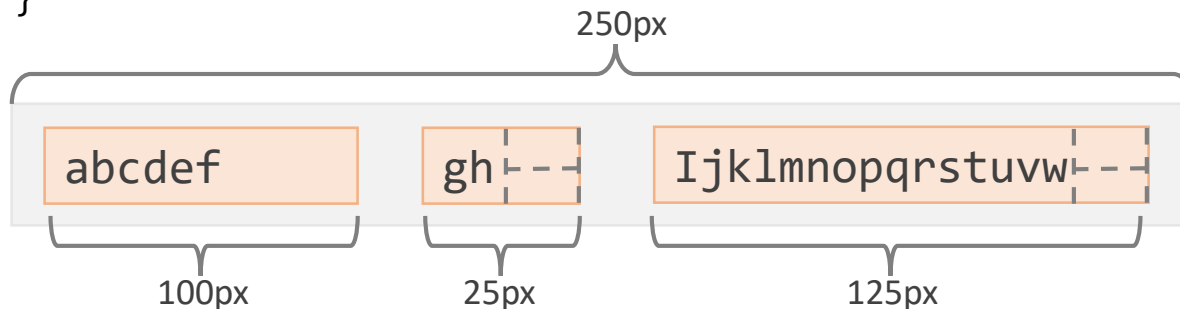
- flex-shrink (기본 값: 1)

```
.container {  
  display: flex;  
  width: 250px  
}
```

```
.item:nth-child(1) {  
  flex-shrink: 0;  
  width: 100px;  
}
```

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

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



# 9. Display Level - Flex Item

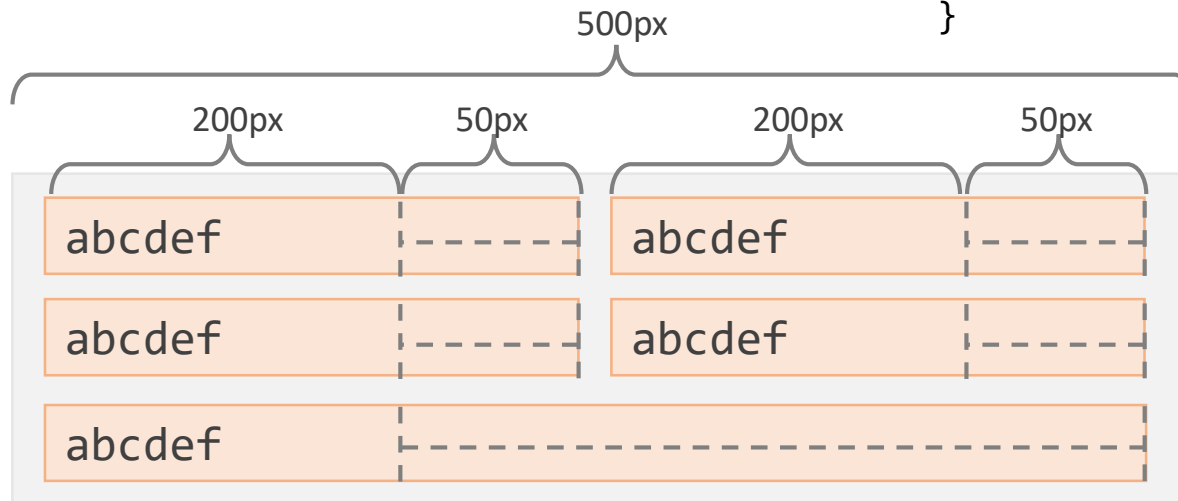
- Flex Item 유연하게 배치하기
  - flex-grow, flex-shrink, flex-basis 복합 사용.

```
.container {  
  display: flex;  
  flex-wrap: wrap;  
  width: 500px  
}
```

```
.item {  
  flex-grow: 1;  
  flex-shrink: 1;  
  flex-basis: 200px;  
}
```

```
.container {  
  display: flex;  
  flex-wrap: wrap;  
  width: 500px  
}
```

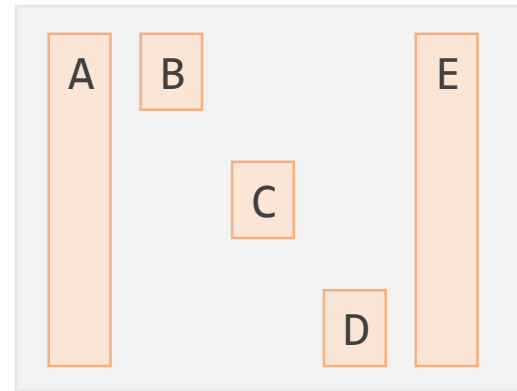
```
.item {  
  flex: 1 1 200px;  
  /*flex-grow: 1;  
  flex-shrink: 1;  
  flex-basis: 200px;*/  
}
```



# 9. Display Level - Flex Item

- Item 수직 축 개별 정렬
  - align-self
    - align-items(flex container 설정)의 item 버전.

```
.container {  
  display: flex;  
  height: 300px;  
}  
  
.item:nth-child(1) {  
  align-self: stretch;  
}  
  
.item:nth-child(2) {  
  align-self: flex-start;  
}  
  
.item:nth-child(3) {  
  align-self: center;  
}  
  
.item:nth-child(4) {  
  align-self: flex-end;  
}
```



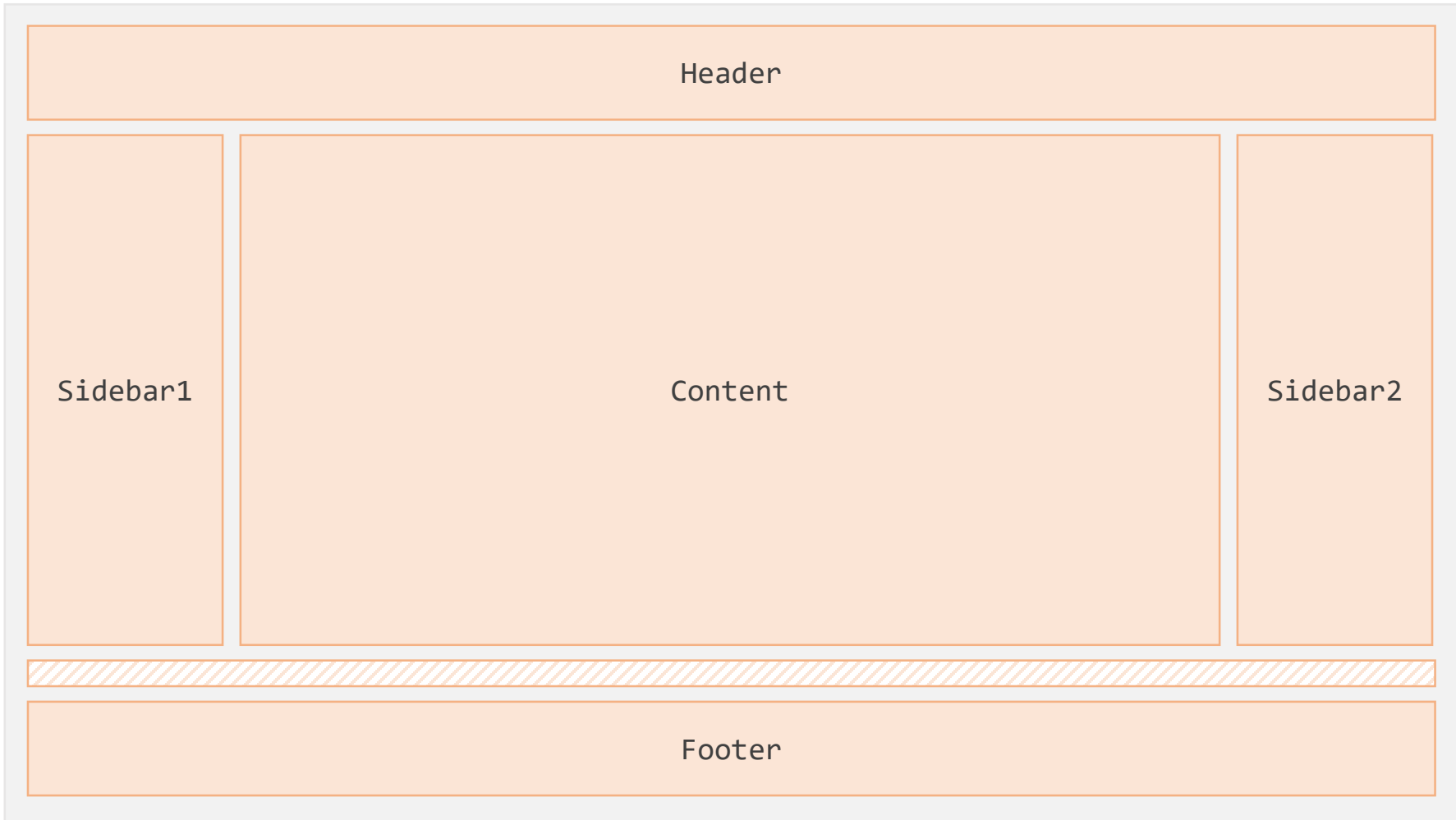


## **9. Display Level – Flex Layout**

# 9. Display Level - Flex Layout

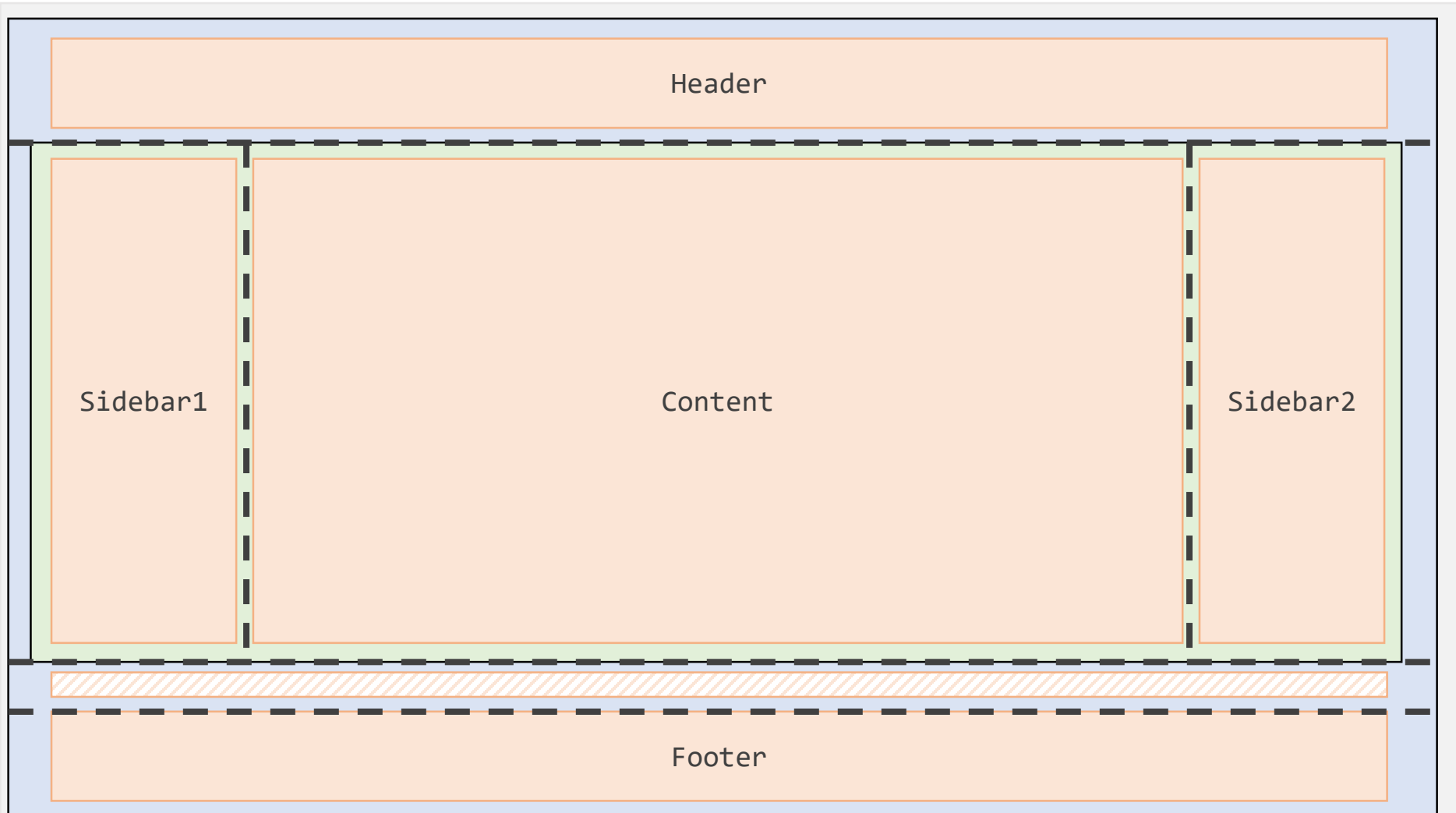
---

- Flex를 이용해 Layout 구성하기



# 9. Display Level - Flex Layout

- Flex를 이용해 Layout 구성하기



# 9. Display Level - Flex Layout

---

- Flex를 이용해 Layout 구성하기 ( 1 / 3 )

```
<html>
  <head>
    <style type="text/css">
      body {
        margin: 0px;
      }

      .container {
        display: flex;
        flex-direction: column;
        min-height: 100%;
      }

      .header {
        flex-shrink: 0;
        height: 100px;
      }

      .body {
        flex-grow: 1;
      }
    </style>
  </head>
</html>
```

# 9. Display Level - Flex Layout

---

- Flex를 이용해 Layout 구성하기 ( 2 / 3 )

```
.empty {  
    flex-shrink: 0;  
    height: 10px;  
}  
  
.footer {  
    flex-shrink: 0;  
    height: 100px;  
}  
  
.main-container {  
    display: flex;  
    flex-direction: row;  
}  
  
.sidebar1, .sidebar2 {  
    flex-shrink: 0;  
    width: 100px;  
}  
  
.content {  
    flex-grow: 1;  
}  
</style>
```

# 9. Display Level - Flex Layout

---

- Flex를 이용해 Layout 구성하기 ( 3 / 3 )

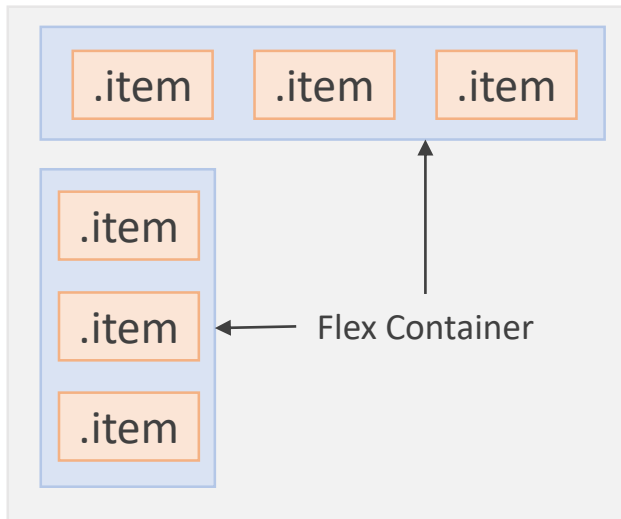
```
</head>
<body>
  <div class="container">
    <div class="header">Header</div>
    <div class="body">
      <div class="main-container">
        <div class="sidebar1">Sidebar1</div>
        <div class="content">Content</div>
        <div class="sidebar2">Sidebar2</div>
      </div>
    </div>
    <div class="empty"></div>
    <div class="footer">footer</div>
  </div>
</body>
</html>
```

## **9. Display Level – Grid Container**

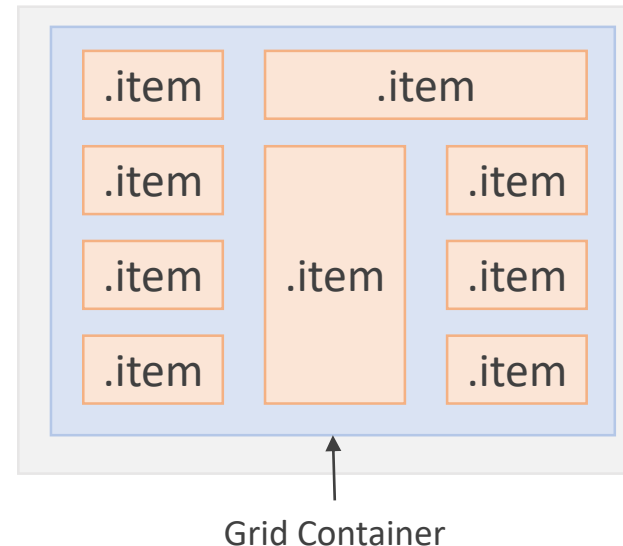
# 9. Display Level – Grid Container

- Display container system
  - Grid
    - Flex는 1차원 레이아웃 (가로 아니면 세로)
    - Grid는 2차원 레이아웃 (가로와 세로)
  - Flex 보다 복잡한 레이아웃을 간편하게 만들 수 있다.

Flex



Grid





# 9. Display Level – Grid Container

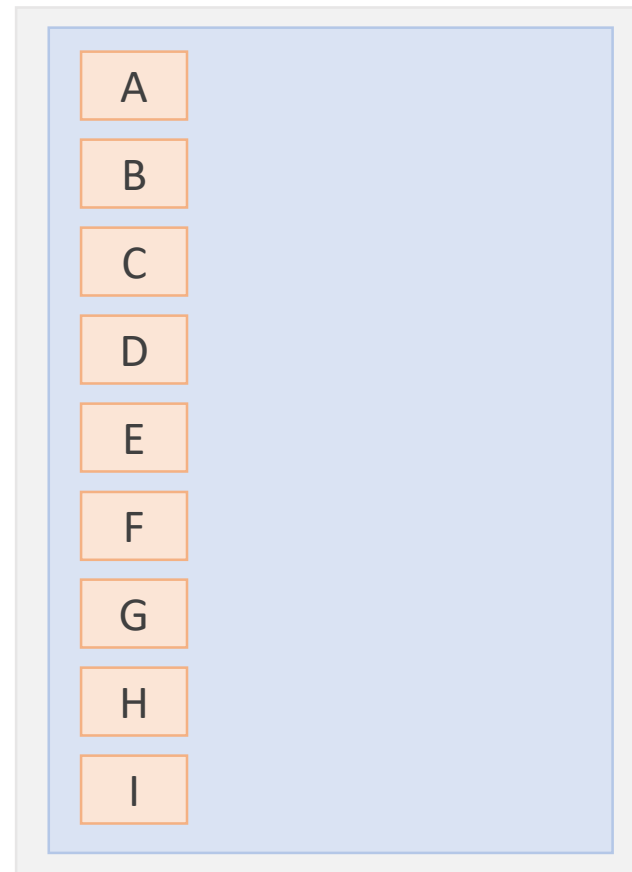
- Flex container system

- Style

```
.container {  
  display: grid;  
  /* display: inline-grid; */  
}
```

```
<div class="container">  
  <div class="item">A</div>  
  <div class="item">B</div>  
  <div class="item">C</div>  
  <div class="item">D</div>  
  <div class="item">E</div>  
  <div class="item">F</div>  
  <div class="item">G</div>  
  <div class="item">H</div>  
  <div class="item">I</div>  
</div>
```

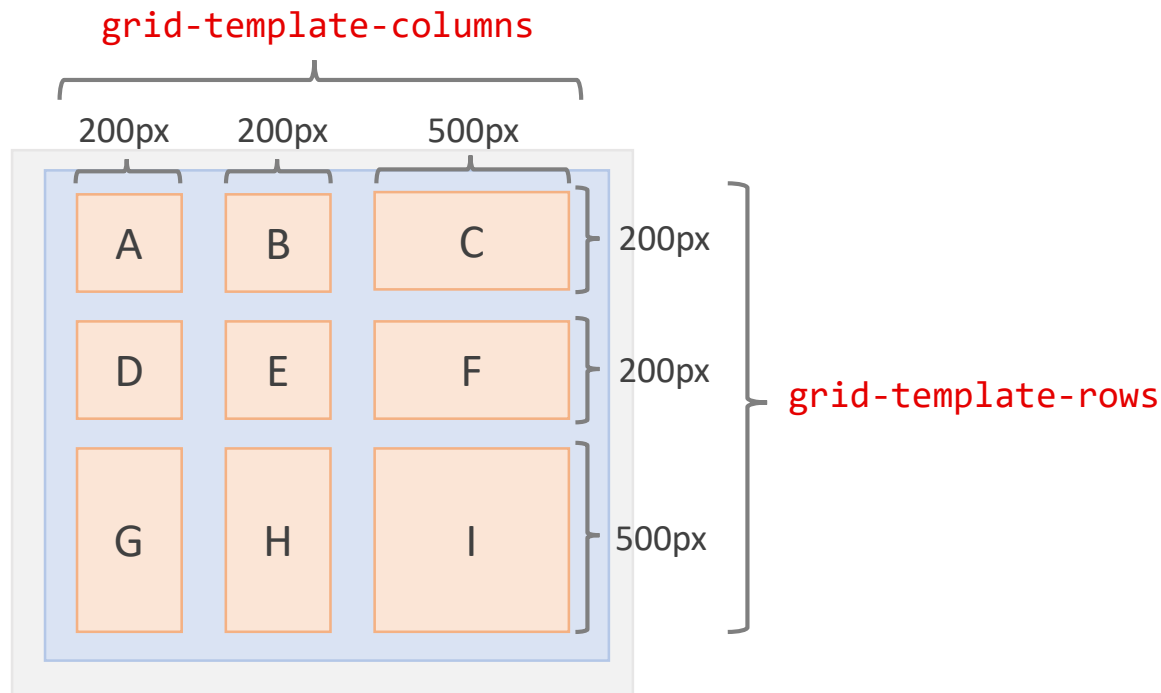
display: grid;  
하나만으로는 아무 일도 일어나지 않는다.



# 9. Display Level – Grid Container

- 그리드 형태 정의 (pixel)
  - grid-template-rows, grid-template-columns

```
.container {  
  display: grid;  
  grid-template-columns: 200px 200px 500px;  
  grid-template-rows: 200px 200px 500px;  
}
```

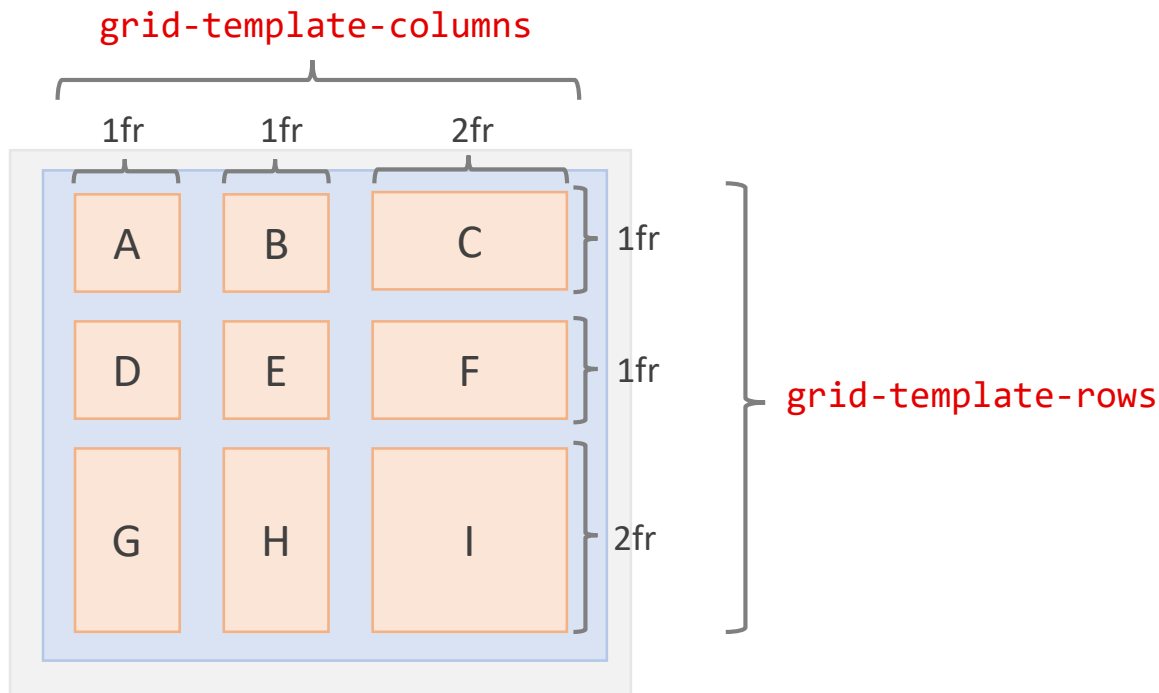


# 9. Display Level – Grid Container

- 그리드 형태 정의 (fraction) – 비율로 정의

- grid-template-rows, grid-template-columns

```
.container {  
  height: 500px;  
  display: grid;  
  grid-template-columns: 1fr 1fr 2fr;  
  grid-template-rows: 1fr 1fr 2fr;  
}
```

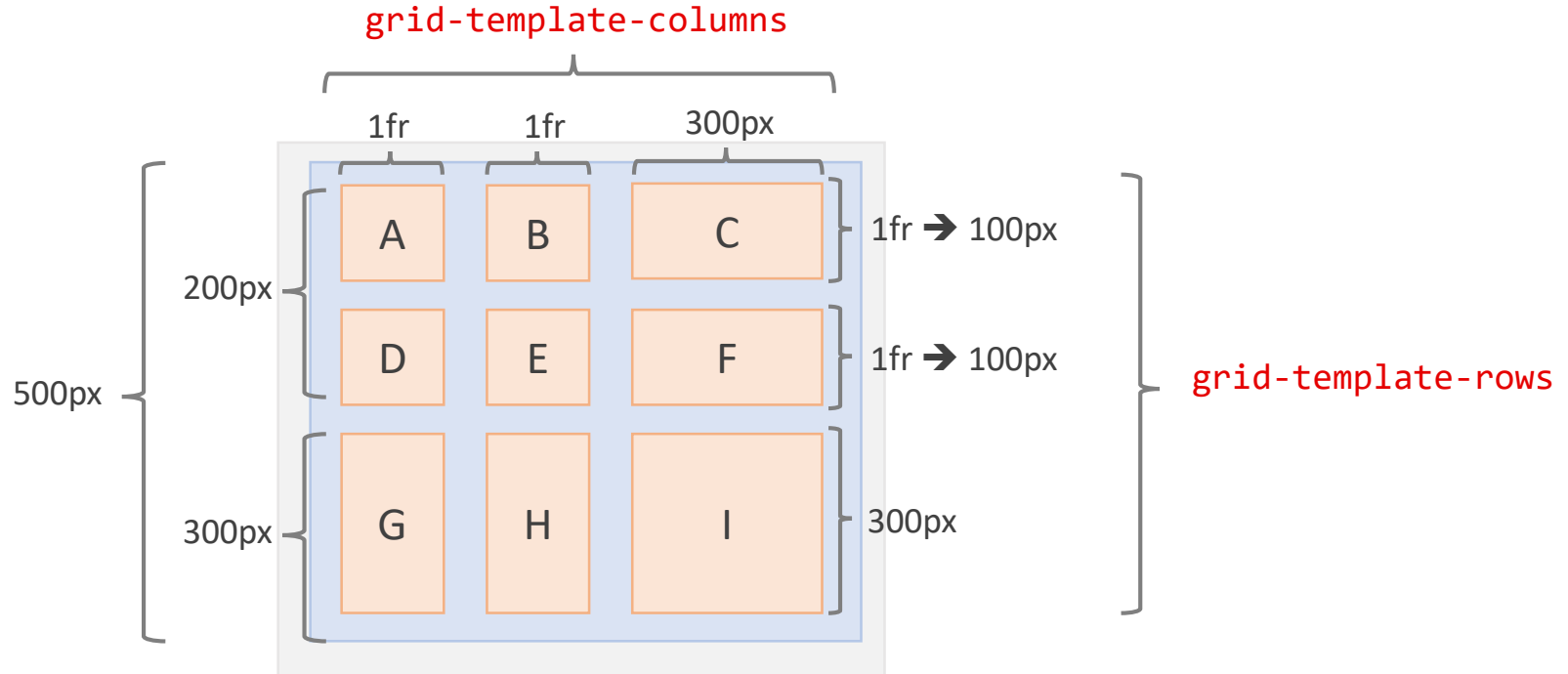


# 9. Display Level – Grid Container

- 그리드 형태 정의 (pixel + fraction) – 비율로 정의

- grid-template-rows, grid-template-columns

```
.container {  
  height: 500px;  
  width: 500px;  
  display: grid;  
  grid-template-columns: 1fr 1fr 300px;  
  grid-template-rows: 1fr 1fr 300px;  
}
```

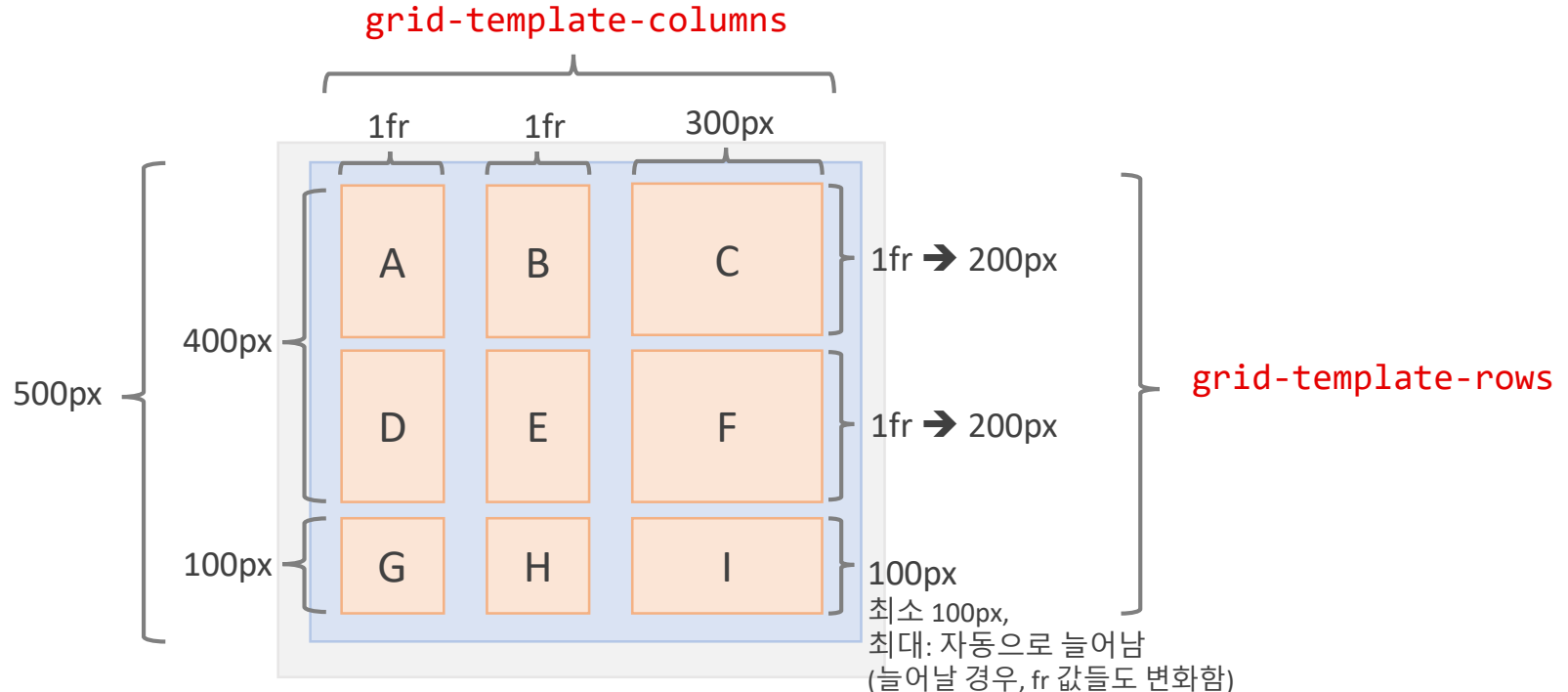


# 9. Display Level – Grid Container

- 그리드 형태 정의 (minmax) - 최소/최대 크기 설정

- grid-template-rows, grid-template-columns

```
.container {  
  height: 500px;  
  width: 500px;  
  display: grid;  
  grid-template-columns: 1fr 1fr 300px;  
  grid-template-rows: 1fr 1fr minmax(100px, auto);  
}
```

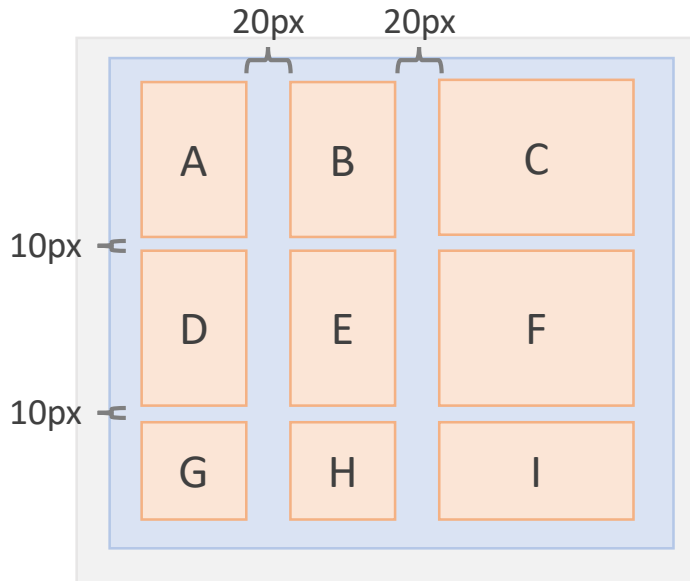


# 9. Display Level – Grid Container

- 간격 만들기

- row-gap, column-gap

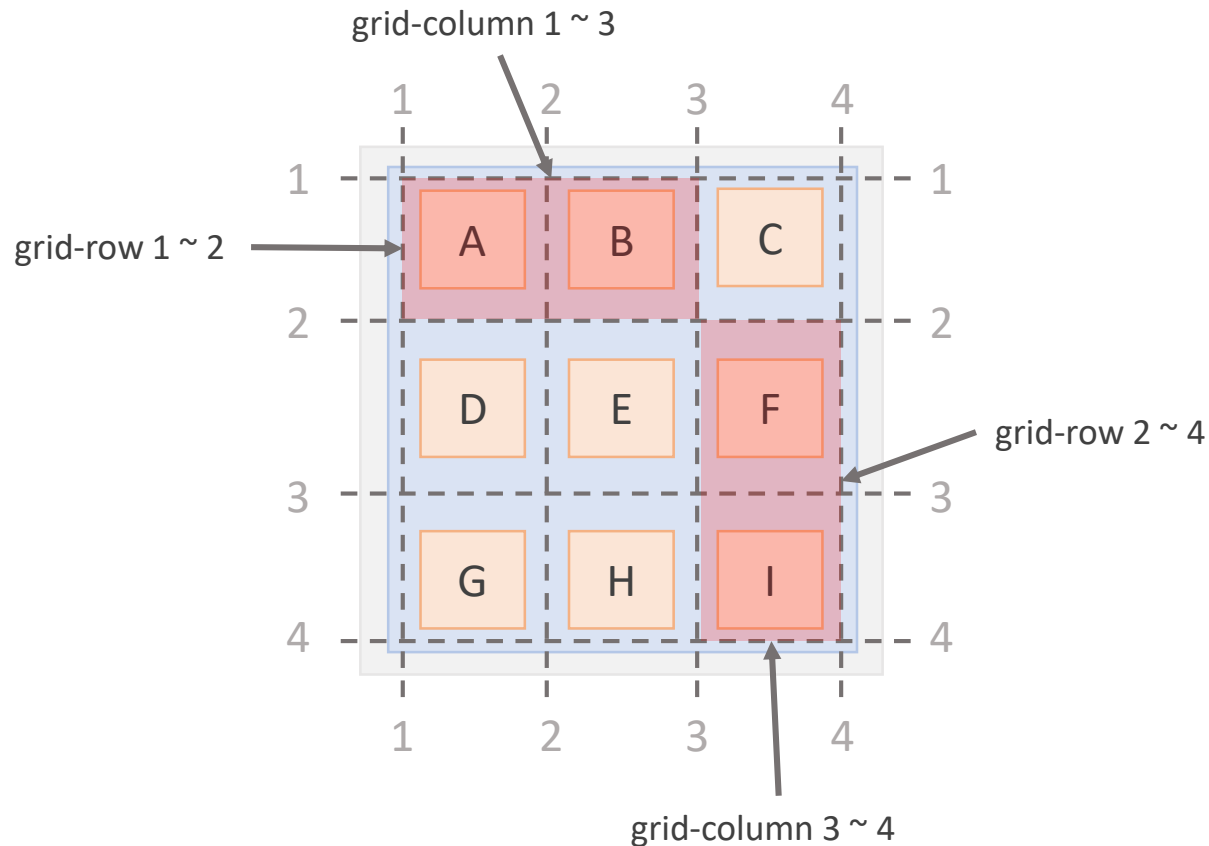
```
.container {  
  height: 500px;  
  display: grid;  
  grid-template-columns: 1fr 1fr 300px;  
  grid-template-rows: 1fr 1fr minmax(100px, auto);  
  row-gap: 10px; /* row의 간격을 10px로 */  
  column-gap: 20px; /* column의 간격을 20px로 */  
}
```



## **9. Display Level – Grid Item**

# 9. Display Level – Grid Item

- 아이템의 영역 지정
  - Grid Item의 영역



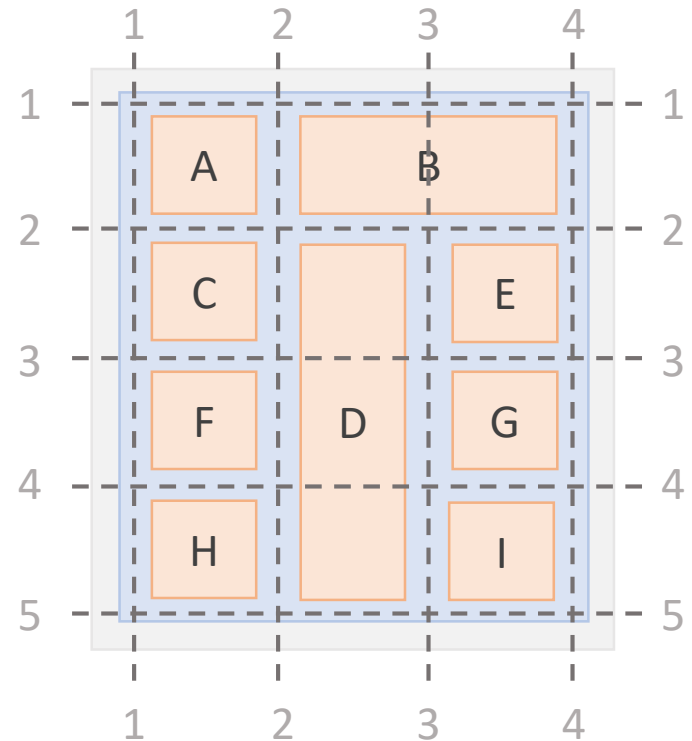


# 9. Display Level – Grid Item

- 아이템의 영역 지정

- grid-column-start, grid-column-end, grid-column, grid-row-start

```
.container {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr;  
  grid-template-rows: 1fr 1fr 1fr;  
  row-gap: 10px;  
  column-gap: 10px;  
}  
.item:nth-child(2) { /* B */  
  display: grid;  
  grid-column-start: 2;  
  grid-column-end: 4;  
  grid-row-start: 1;  
  grid-row-end: 2;  
  background-color: antiquewhite ;  
}  
.item:nth-child(4) { /* D */  
  display: grid;  
  grid-column-start: 2;  
  grid-column-end: 3;  
  grid-row-start: 2;  
  grid-row-end: 5;  
  background-color: aqua ;  
}
```



# 9. Display Level – Grid Item

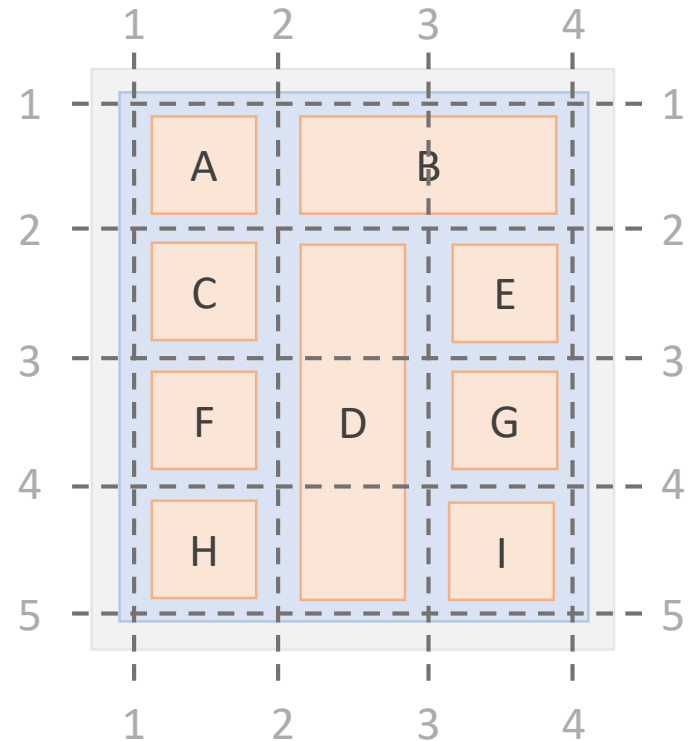
- 아이템의 영역 지정

- grid-column-start, grid-column-end, grid-column, grid-row-start

```
.container {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr;  
  grid-template-rows: 1fr 1fr 1fr;  
  row-gap: 10px;  
  column-gap: 10px;  
}
```

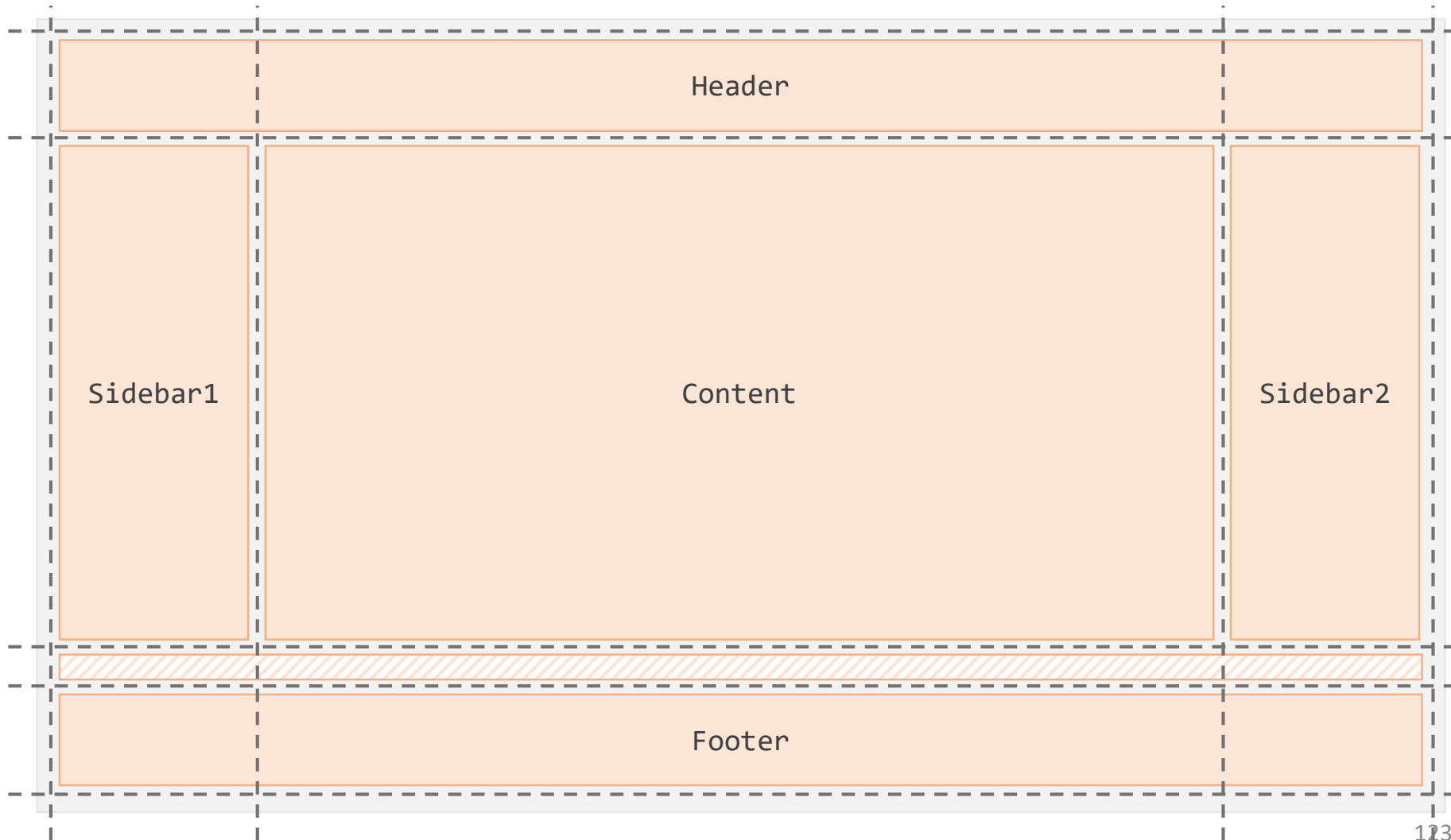
```
.item:nth-child(2) { /* B */  
  display: grid;  
  grid-column: 2 / 4;  
  grid-row: 1 / 2;  
  background-color: antiquewhite ;  
}
```

```
.item:nth-child(4) { /* D */  
  display: grid;  
  grid-column: 2 / 3;  
  grid-row: 2 / 5;  
  background-color: aqua ;  
}
```



# 9. Display Level - Grid Layout

- Grid를 이용해 Layout 구성하기



# 9. Display Level - Grid Layout

---

- Grid를 이용해 Layout 구성하기 ( 1 / 2 )

```
<html>
  <head>
    <style type="text/css">
      body {
        margin: 0px;
      }
      .container {
        min-height: 100%;
        display: grid;
        grid-template-columns: 100px 1fr 100px;
        grid-template-rows: 200px 1fr 10px 200px;
        row-gap: 10px;
        column-gap: 10px;
      }
      .container > .item {
        display: grid;
      }
      .header, .footer, .empty {
        grid-column: 1 / 4;
      }
    </style>
  </head>
```

# 9. Display Level - Grid Layout

---

- Grid를 이용해 Layout 구성하기 ( 2 / 2 )

```
<body>
  <div class="container">
    <div class="item header">Header</div>
    <div class="item">Sidebar1</div>
    <div class="item">Content</div>
    <div class="item">Sidebar2</div>
    <div class="item empty"></div>
    <div class="item footer">Footer</div>
  </div>
</body>
</html>
```

# 10. Position

# **10. Position - position**

# 10. Position - position

- 엘리먼트의 위치를 지정하는 속성

## Position Values

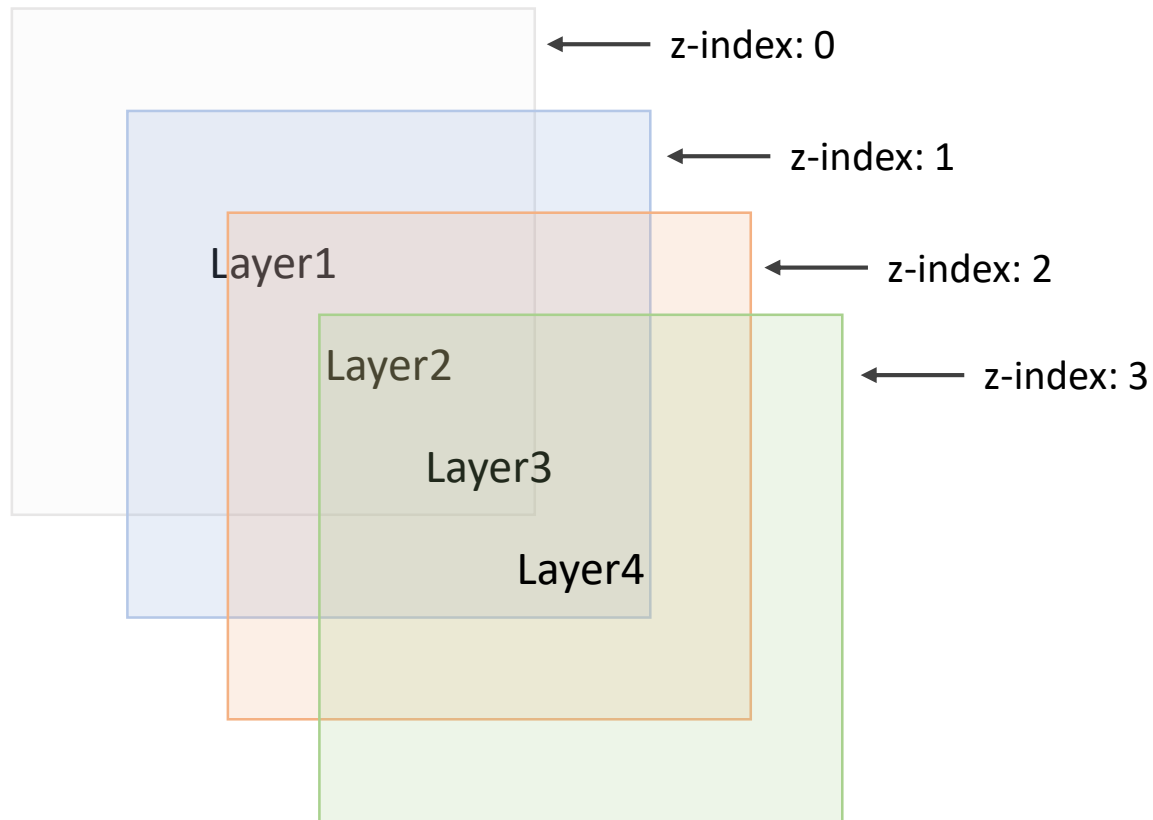
Position Value	Description
static	아무런 효과가 없음. 정상적인 흐름으로 엘리먼트가 배치됨
relative	현재 위치를 기준으로 좌표가 시작되며, top, left, right 좌표가 만들어짐.
fixed	스크롤을 하더라도 항상 같은 위치에 배치됨. top, right, bottom, left 속성으로 배치함.
absolute	절대 위치 지정. top, right, bottom, left로 배치. 상위 엘리먼트가 relative인 경우 상대위치로 배치됨.
sticky	정상적인 흐름으로 엘리먼트가 배치되었다가 스크롤이 발생할 경우, 최상단으로 고정됨.



# **10. Position – z-index**

# 10. Position – z-index

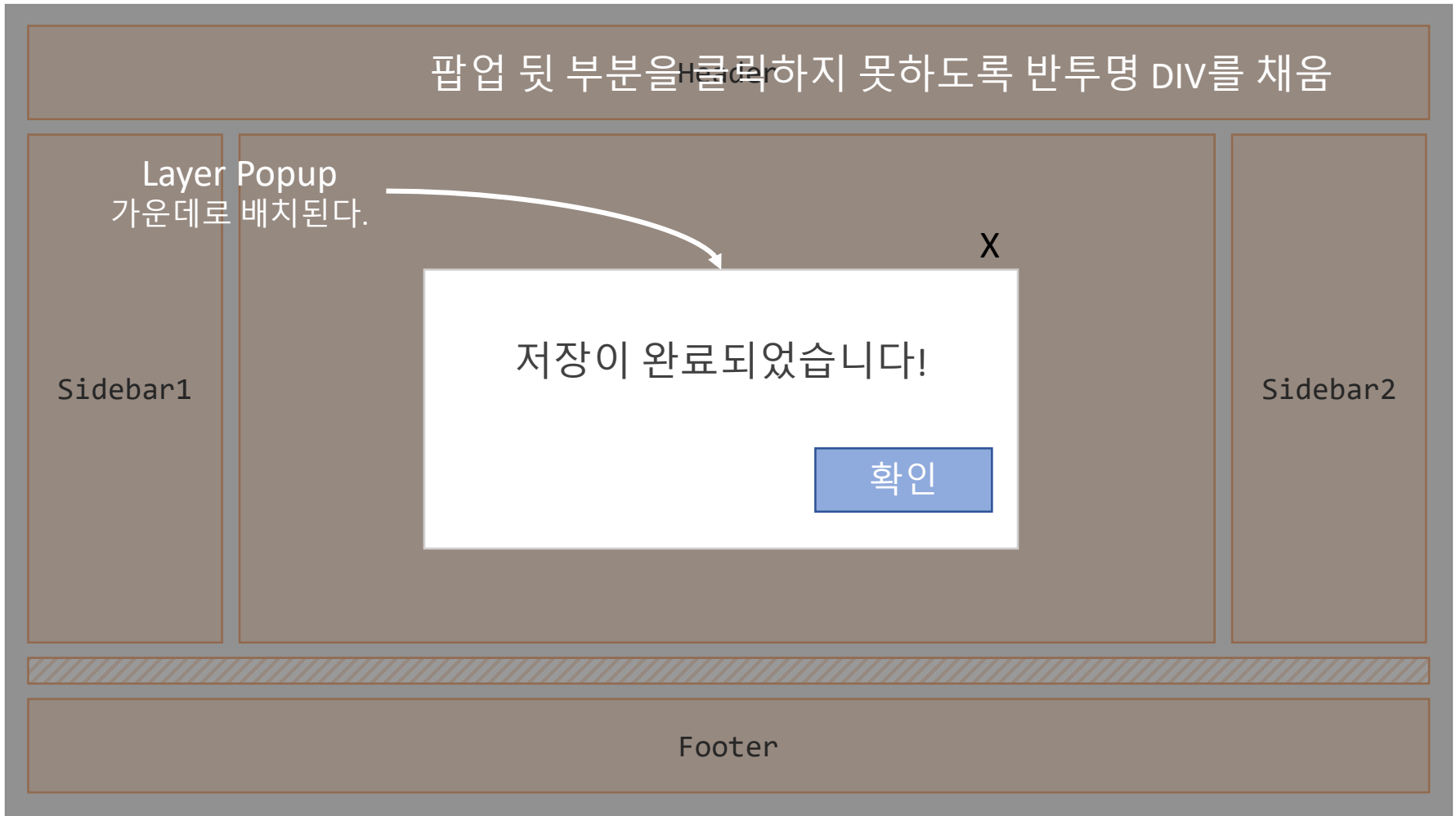
- 엘리먼트를 Layer 처럼 겹쳐서 표현함.
  - position 속성과 함께 사용됨.
  - 일반적으로 Layer Popup 등을 만들 때 사용한다.



# **10. Position – Layer Popup**

# 10. Position – Layer Popup

- position과 z-index를 이용해 Layer Popup 만들기



# 10. Position – Layer Popup

---

- position과 z-index를 이용해 Layer Popup 만들기 ( 1 / 3 )

```
.pointer { cursor: pointer; }
div.backdrop {
  background-color: #000000;
  opacity: 0.8;
  z-index: 1;
  position: fixed;
  top: 0;
  left: 0;
  bottom: 0;
  right: 0;
}
div.layerpopup-wrapper {
  display: flex;
  align-items: center;
  justify-content: center;
  padding: 0px;
  z-index: 2;
  position: fixed;
  top: 0;
  left: 0;
  bottom: 0;
  right: 0;
}
```

# 10. Position – Layer Popup

---

- position과 z-index를 이용해 Layer Popup 만들기 ( 2 / 3 )

```
div.layerpopup {  
    background-color: #FFFFFF;  
}  
div.layerpopup > .layerpopup_header {  
    padding-right: 10px;  
    padding-top: 10px;  
    text-align: right;  
}  
div.layerpopup > .layerpopup_content {  
    padding: 10px;  
    max-width: 500px;  
    max-height: 400px;  
}
```

# 10. Position – Layer Popup

- position과 z-index를 이용해 Layer Popup 만들기 ( 3 / 3 )

```
<body>
  ... 생략 ...

  <div class="backdrop"></div>
  <div class="layerpopup-wrapper">
    <div class="layerpopup">
      <div class="layerpopup_header">
        <span class="pointer">X</span>
      </div>
      <div class="layerpopup_content">
        <p>저장이 완료되었습니다!</p>
        <p style="text-align: right;">
          <button>확인</button>
        </p>
      </div>
    </div>
  </div>
</body>
```

# 10. Position – Layer Popup

- position과 dialog를 이용해 Layer Popup 만들기





# 10. Position – Layer Popup

---

- position과 dialog를 이용해 Layer Popup 만들기 ( 1 / 2 )

```
.pointer { cursor: pointer; }
dialog.layerpopup::backdrop {
    background-color: #000000;
    opacity: 0.8;
}
dialog.layerpopup {
    padding: 0px;
    position: fixed;
    top: 0;
    left: 0;
    bottom: 0;
    right: 0;
}
dialog.layerpopup > .layerpopup_header {
    padding-right: 10px;
    padding-top: 10px;
    text-align: right;
}
dialog.layerpopup > .layerpopup_content {
    padding: 10px;
    max-width: 500px;
    max-height: 400px;
}
```

# 10. Position – Layer Popup

---

- position과 dialog를 이용해 Layer Popup 만들기 ( 2 / 2 )

```
<body>
  ... 생략 ...
  <dialog class="layerpopup">
    <div class="layerpopup_header">
      <span class="pointer">X</span>
    </div>
    <div class="layerpopup_content">
      <p>저장이 완료되었습니다!</p>
      <p style="text-align: right;">
        <button>확인</button>
      </p>
    </div>
  </dialog>
</body>
```

- 개발자 도구(F12) – 콘솔에서 입력
  - var popup = document.getElementsByClassName("layerpopup")
  - popup[0].showModal();
  - popup[0].close();

# 11. Overflow

# 11. Overflow

- width, height가 설정된 엘리먼트의 범위보다 콘텐츠가 더 클 때 콘텐츠를 어떻게 표현할 지 정의함.

## Overflow Values

Overflow Value	Description
visible	기본값. 엘리먼트 바깥으로 콘텐츠가 빠져나옴.
hidden	엘리먼트 바깥으로 빠져나온 콘텐츠가 숨겨짐
scroll	스크롤바가 생김
auto	scroll과 유사하지만, 필요에 따라 스크롤바가 생김

overflow-x Value	Description
visible	기본값. 엘리먼트 바깥으로 콘텐츠가 빠져나옴.
hidden	엘리먼트 바깥으로 빠져나온 콘텐츠가 숨겨짐
scroll	수평 스크롤바가 생김
auto	scroll과 유사하지만, 필요에 따라 수평 스크롤바가 생김

overflow-y Value	Description
visible	기본값. 엘리먼트 바깥으로 콘텐츠가 빠져나옴.
hidden	엘리먼트 바깥으로 빠져나온 콘텐츠가 숨겨짐
scroll	수직 스크롤바가 생김
auto	scroll과 유사하지만, 필요에 따라 수직 스크롤바가 생김

# **12. Transform, Transition**

# 12. Transform

# 12. Transform

- Transform 속성을 사용하면 다음과 같은 2D 변형 방법을 사용할 수 있다.

CSS 2D Transform Methods

Function	Description
<code>translate(x,y)</code>	X축과 Y축을 따라 요소를 이동하는 2D 변환을 정의
<code>translateX(n)</code>	X축을 따라 요소를 이동하는 2D 변환을 정의합니다
<code>translateY(n)</code>	Y축을 따라 요소를 이동하는 2D 변환을 정의합니다
<code>scale(x,y)</code>	요소의 너비와 높이를 변경하는 2D 축척 변환을 정의
<code>scaleX(n)</code>	요소의 폭을 변경하는 2D 축척 변환을 정의
<code>scaleY(n)</code>	요소의 높이를 변경하는 2D 축척 변환을 정의
<code>rotate(angle)</code>	2D 회전을 정의하고 각도는 매개변수에 지정

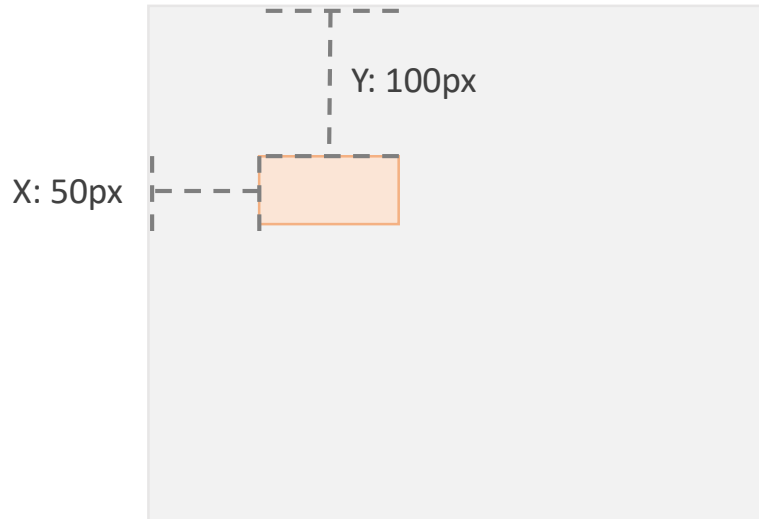
- 아래 처럼 여러 속성을 동시에 사용할 수 있다.
  - `transform: translate(x, y) scale(x, y) rotate(angle);`

# 12. Transform

- Translate

- x축과 y축에 지정된 매개변수에 따라 요소를 현재 위치에서 이동시킴.

```
div {  
  width: 300px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
  transform: translate(50px, 100px);  
  /*transform: translateX(50px) translateY(100px);*/  
}
```





# 12. Transform

---

- Scale
  - 너비와 높이에 지정된 매개변수에 따라 요소의 크기를 늘리거나 줄임.
    - 아래 이미지로 테스트
    - <https://interactive-examples.mdn.mozilla.net/media/examples/firefox-logo.svg>

```
img {  
  margin: 150px;  
  width: 100px;  
  height: 100px;  
}  
img.scale1 { transform: scale(0, 0); }  
img.scale2 { transform: scale(1, 1); }  
img.scale3 { transform: scale(2, 2); }  
img.scale4 { transform: scale(0.5, 0.5); }  
img.scale5 { transform: scale(2, 3); }  
img.scale6 { transform: scale(-1, 1); }  
img.scale7 { transform: scale(1, -1); }
```

# 12. Transform

---

- Rotate
  - 주어진 각도에 따라 요소를 시계 방향 또는 시계 반대 방향으로 회전시킴.
    - 아래 이미지로 테스트
    - <https://interactive-examples.mdn.mozilla.net/media/examples/firefox-logo.svg>

```
img {  
  margin: 150px;  
  width: 100px;  
  height: 100px;  
}  
img.rotate1 { transform: rotate(45deg); }  
img.rotate2 { transform: rotate(90deg); }  
img.rotate3 { transform: rotate(135deg); }  
img.rotate4 { transform: rotate(180deg); }  
img.rotate5 { transform: rotate(225deg); }  
img.rotate6 { transform: rotate(270deg); }  
img.rotate7 { transform: rotate(315deg); }  
img.rotate8 { transform: rotate(-45deg); }
```

# 12. Transition

# 12. Transition

---

- Transition을 사용하면 지정된 기간 동안 속성 값을 부드럽게 변경할 수 있다.
- 빨간색 Div에 마우스를 올리면 width를 0.2초 동안 변경하는 예제

```
div {  
  width: 100px;  
  height: 100px;  
  background: red;  
  transition: width 0.2s;  
}
```

```
div:hover {  
  width: 300px;  
}
```

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

# 12. Transition

---

- 빨간색 Div에 마우스를 올리면 width와 height를 0.2초 동안 변경하는 예제

```
div {  
  width: 100px;  
  height: 100px;  
  background: red;  
  transition: width 0.2s, height 0.2s;  
}
```

```
div:hover {  
  width: 300px;  
  height: 300px;  
}
```

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

# 감사합니다.

---

HTML5 & CSS3

장민창

mcjang@hucloud.co.kr